'68' MICRO JOURNAL

$2.95USA

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $29.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME VI ISSUE VIII • Devoted to the 68XX User • Aug./Sept. 1984

"Small Computers Doing Big Things"

PHOTO CREDIT: NASA

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

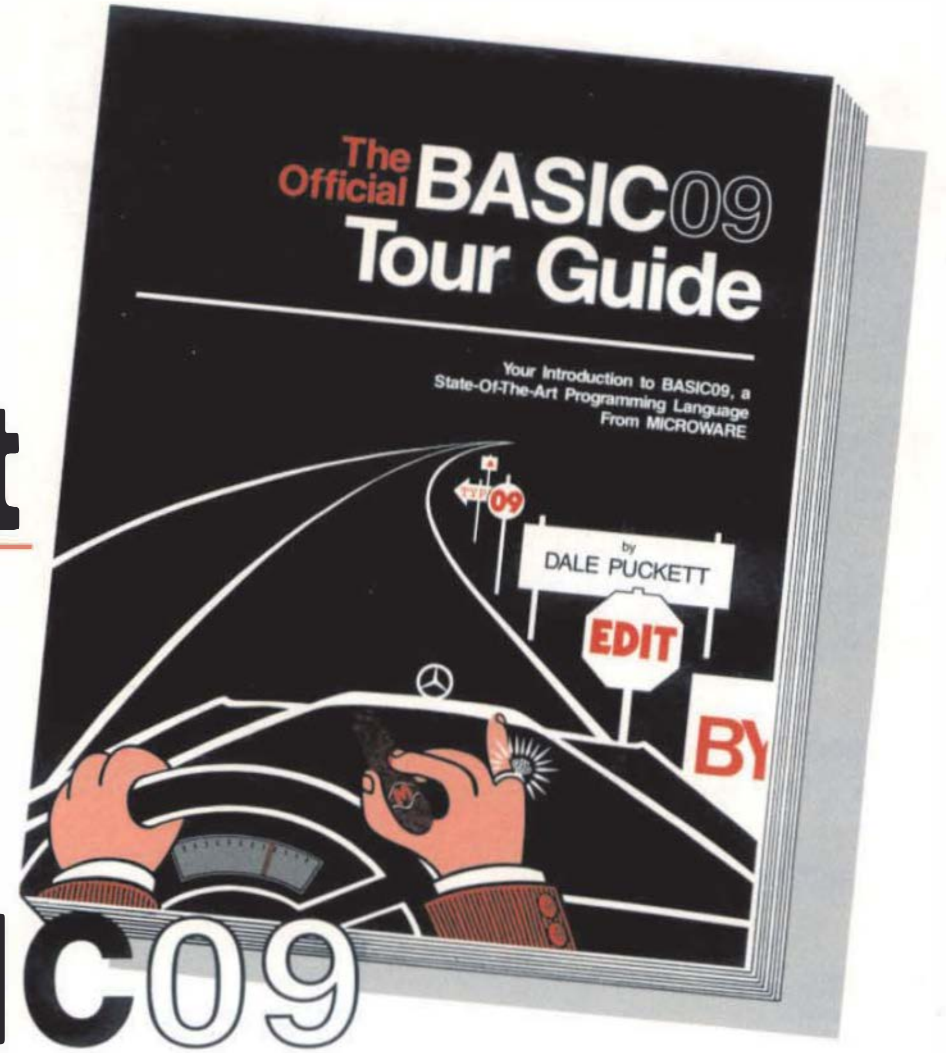
Up to Three Terminals - Instant expansion

*Trademark of Technical Systems Consultants

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Get the most out of BASIC09


The Official BASIC09 Tour Guide
Your Introduction to BASIC09, a State-Of-The-Art Programming Language From MICROWARE
09
by DALE PUCKETT
EDIT
BY

The **OFFICIAL BASIC09 TOUR GUIDE** is skillfully written in a friendly and easy-to-read style. Just perfect for those new to computers and to BASIC09. It's also a *valuable reference book* for programmers, engineers, students and hobbyists, providing an in-depth look at BASIC09 plus an overview of the OS-9 operating system. Comprehensive reference sections on BASIC09 and OS-9 commands are also included.

The book "maps" out your route through the Mercedes of Basics . . . BASIC09 and puts you in the driver's seat in no time. Fasten your seatbelt, sit back and enjoy the ride to perfecting your programming skills.

MICROWARE . . .

The **OFFICIAL BASIC09 TOUR GUIDE** comes from the people who wrote BASIC09. As the leader in 6809 system software, we at MICROWARE care about our users and want to help you get the most from our products.

It's Easy to Order.

Phone orders are accepted from MasterCard or VISA cardholders or for COD shipment. You can also order by mail using the coupon below. Quantity discounts are available to educational organizations and dealers. For further information contact Microware.

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322
Telephone 515/224-1929
Telex 910-520-2535

Please send ________ copies of the **Basic09 Tour Guide** book at $18.95 each. Add $2.00 for UPS shipping in the U.S. or $5.00 for overseas air mail per book. Iowa residents add 4% sales tax.

Name ____________________

Address ____________________

City ____________________

State __________ Zip __________

□ I have enclosed a check

□ Charge to my bank card:
MasterCard □ VISA □
Card Number __________
Expiration __________

microware®

Specialists in system software for 68-family microprocessors since 1977.

OS-9 and BASIC09 are trademarks of Microware and Motorola.

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer
-
GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor
-
Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Vol.VI,Issue VIII Aug./Sept.84

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM**, **I/O**, etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX** III systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III, a 256K CMOS Static RAM Board (#72)**, and a **3-port Intelligent Serial I/O Processor (#11)**.

The **GMX 6809** CPU III can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track DSDD drives	**$6198.79**
with **#88** dual 8" DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8" DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The #39 systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8" DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your **GIMIX** system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1698.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

TRADE UP YOUR CoCo!

GIMIX will allow you up to $1100.00 credit toward the purchase of any GIMIX system when you trade-in your working Color Computer, peripherals, and original software. The trade-in value is limited to 110% of the RADIO SHACK™ list price at the time your order is placed. You pay the freight. This offer is good only in the Continental U.S.; is limited to the first 100 orders; and expires on 9/30/84. Only one trade-in per customer.

GIMIX inc.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

©1984 GIMIX, INC. 4-84

Microware presents 4 new OS-9 software packages.

1 LEVEL II PRINT SPOOLING SYSTEM

This versatile package gives your OS-9 Level Two System a complete print spooling management capability for time-sharing applications. Features of the spooling system are:

- *Handles up to seven independent spooling devices and queues with "print on first available device" feature.*
- *Prints large block header pages between listings with date, time, user name and job name.*
- *Multiple listing copy option.*
- *Complete forms change capability for each job and device.*
- *Prints formatted or unformatted listings.*
- *Status command displays print queues and status.*
- *User can kill or change priority of queued jobs.*

*Available **only** for OS-9 Level Two Systems.*

Suggested List Price: $150.00 Manual Only: $15.00

2 RMA RELOCATABLE MACRO ASSEMBLER

At last — a full feature relocatable macro assembler and linkage editor for OS-9. RMA permits sections of assembly language programs to be independently assembled to "relocatable object files". The linkage editor takes any number of program sections and/or library sections and combines them into a single executable OS-9 memory module. Global data (including indexed and direct addressing modes) and program references are automatically resolved in the process. The macro facility permits commonly used statement sequences to be defined, then used within the program with appropriate parameter substitution. RMA also supports conditional assembly and library source files.

Suggested List Price: $200.00 Manual Only: $20.00

3 OS-9 FILE HANDLER TOOLBOX

Introducing a special toolbox for OS-9 users who do a lot of file manipulation! A collection of 12 useful OS-9 command programs; Most can be used as "filters" using OS-9 pipeline facilities. Included are:

***D** — unformatted directory listing with "wild card" matching*
***Compress** — does character compression on text files.*
***Expand** — restores a "compressed" file to the original state.*
***Split** — breaks a file into smaller files.*
***Space** — indents lines with optional spacing between lines.*
***Code** — decodes any key on a keyboard to hex.*
***Qsort** — quick sort for small files, directories, etc.*
***Pr** — versatile formatted file printing utility.*
***Tr** — transliterates text pattern to substitution pattern.*
***Grep** — searches file for a pattern and prints matching lines.*
***Xmode** — same "tmode" except changes are made to the device descriptor.*
***Count** — counts words, lines, or characters within a text file.*

Suggested List Price $85.00

4 ENTERTAINMENT PACK I

A collection of games and other interesting programs that are not only entertaining but serve as good instructional examples of Basic09 programming techniques. All programs include complete Basic09 source files and can be easily edited to run on standard alphanumeric or graphics terminals.

***Blkjak** — A Vegas-rules blackjack game.*
***Clk** — graphical display of a wall clock on your terminal.*
***Dogs** — Greyhound racing with simulated graphics.*
***Eliza** — Basic09 version of the famous artificial intelligence simulation of natural language dialogue with a psychiatrist.*
***Haiku** — Program that creates original "haiku" prose.*
***Quest** — a mini-"Adventure" game.*
***Rats** — find your way out of a computer-generated maze — from a rat's point of view.*
***Towers** — a graphical display of the solution to the "Tower of Hanoi" puzzle.*

Suggested List Price: $85.00

MICROWARE®

Microware Systems Corporation
P.O. Box 4865 • Des Moines, IA 50304
515-279-8844 • Telex 910-520-2535

OS-9 and Basic09 are trademarks of Microware and Motorola.
Unix is a trademark of Bell Labs.

$4,325
FOR A WORLD-CLASS SS-50 COMPUTER

Smoke Signal's VAR/68™ gives you:

- Fabled Chieftain performance that led the pack in tough Benchmark surveys
- Integrated, easy-to-use software that covers your *complete* business needs
- Proven reliability backed by our exclusive Endurance-Certification Program
- Extremely good looks and unsurpassed operator comfort

$4,325: A PRICE CALCULATED TO GET YOU HOOKED ON THIS BLOCKBUSTER SS-50

That price buys you a VAR/68 computer with multi-user, multi-tasking capabilities, and an ergonomically designed terminal. You get 128K RAM—expandable to 1mb. Eight serial ports, up to 16 if desired. Two parallel ports—and more are available. Plus a long list of other impressive capabilities.

Smoke Signal's experience allows us to offer OS-9 and other UNIX-like, and multi-user operating systems.

The styling is completely new—fashioned for the utmost in operator comfort. And it's remarkably compact. VAR/68 is a combination of great performance and good looks demanded by the office of today.

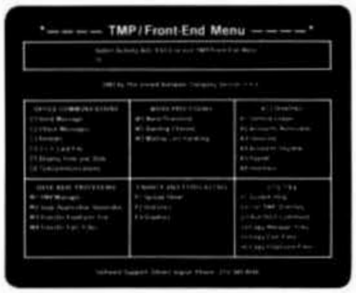
VAR/68 IS TOUGH, BUT SMOKE SIGNAL GIVES YOU *EXTRA* PROTECTION

(1) Your new computer is Endurance-Certified before delivery. That's an exclusive quality-assurance process that guarantees perfect operations from day one. (2) Our Advance-Replacement program is yours for a low fixed charge. (3) You get instant diagnostic service by telephone. It's free. (4) Normal repairs are handled with super speed. (5) Software *and* hardware support are part of doing business with Smoke Signal.

TOTAL INTEGRATED SOFTWARE GIVES YOUR BUSINESS *SOLUTIONS* INSTEAD OF PROBLEMS

Powerful business application programs are ingeniously interlinked to give even untrained operators a quick, smooth upper hand. The VAR/68 is a joy for first-time users, and an unprecedented productivity tool for anyone who wants new dimensions of control over critical business matters.

This screen tells part of the story:



GET A BIG DISCOUNT ON YOUR INITIAL ORDER

Most re-sellers can save up to 42 percent—even on small orders. Smoke Signal's price schedule is a powerful profit-maker for dealers of almost every description.

CALL SMOKE SIGNAL OR WRITE FOR MORE INFORMATION ON THE VAR/68 COMPUTER FAMILY

VAR/68™
BY SMOKE SIGNAL

The computer designed for your specialized application software

SMOKE SIGNAL
Products and Support for VARs

31336 Via Colinas • Westlake Village, CA 91362-3984 • (818) 889-9340

THE 68000 FROM SMOKE SIGNAL!

ADD 68000 AND UNIX™ * TO YOUR EXISTING SS-50 COMPUTER AT PRICES 50% TO 75% OFF LIST

THANK YOU

Seven years ago, Smoke Signal was founded to sell state-of-the-art computer products, by mail, to individual professional programmers and hardware engineers. At that time, most big companies did not believe in the power or future of microcomputers for serious computing applications. Only after you, the individual computer user, proved the viability of the microcomputer was Smoke Signal able to sell systems for business uses. However, as we progressed to become the leader in SS-50 systems, we had to add the sales and technical support services demanded by these business customers — and our prices for complete systems reflected these added costs.

With the introduction of our 68000 products, we wanted to find a way to say thanks to you, our original customers, the individual computer users, and still offer complete sales and technical support to our business customers for complete systems. We think this offer accomplishes both of these goals. We are offering you a choice of upgrade kits that will bring any SS-50 computer up to the electrical equivalent of our complete 68000 computer systems at prices far below complete system prices. In fact, the prices offered are 50% or more off our normally low prices for the components contained in the upgrade kits.

This special offer is limited to one upgrade kit per customer and is our way of saying thanks to those of you who had confidence in us from the beginning.

THE UPGRADES

The following upgrade kits were designed so that any SS-50 system can be upgraded to 68000/UNIX.

SWTP UPGRADE. $2,800.00

Contains: LMB-1A SS-50C Motherboard, DCB-4A floppy controller, PSA-1 Winchester/Tape DMA interface, SCB-68K 68000 CPU, SER-2 dual serial board, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

GIMIX UPGRADE. $2,500.00

Contains: Same as SWTP Upgrade except allows you to use your GIMIX motherboard, serial board and Winchester power supply.

Users of standard SMOKE SIGNAL systems may choose one of the followingupgrade kits:

For SSB floppy based systems:

SS-FD UPGRADE $2,100.00

Contains: SCB-68K 68008 CPU, PSA-1 Winchester/Tape OMA interface, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

For SSB Winchester based systems:

SS-HD UPGRADE. $500.00

Contains: SCB-68K 68008 CPU and REGULUS.

COMPLETE SYSTEMS

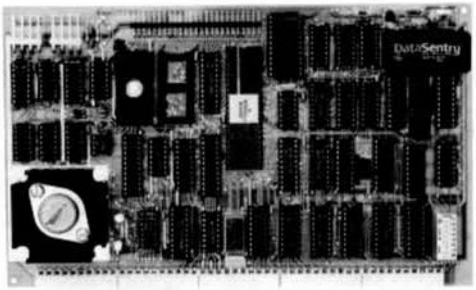
SMOKE SIGNAL is also making available complete VAR/68K™ systems at dramatic discounts. This offer is only available through SMOKE SIGNAL dealers. Contact SMOKE SIGNAL directly for information about how to order a complete VAR/68K system.

RULES OF THE OFFER

1) Limit, one upgrade system per customer.
2) Prices valid through December 31, 1984.
3) Orders must be accompanied by full payment in the form of individual check or credit card authorization.
4) Support will only be provided for systems containing the following SMOKE SIGNAL boards: SCB-68K, DCB-4A, PSA-1, and a motherboard such as the LM8-1A with extended addressing and main terminal I/O at FF7E8.
5) While we feel that most static RAM boards will work with these upgrades, we only guaranty compatibility with systems containing SMOKE SIGNAL static or dynamic RAM.

VAR/68K is a trademark of Smoke Signal.
REGULUS is a registered trademark of Alcyon Corp.; UNIX is a registered trademark of Bell Laboratories; OS9 and OS9/68K are trademarks of Microware; MACSBUG is a trademark of Motorola Inc.

***Regulus the OS offered is UNIX Compatible**



PRODUCTS

The heart of all these upgrade kits is SMOKE SIGNAL'S new SCB-68K 8 MHz 68008 CPU Board. This standard (5½" x 9") board will replace a SCB-69 CPU Board in any SMOKE SIGNAL computer with current revision boards. This board contains a real-time clock with battery back-up, 2 EPROM slots for up to 64K bytes of storage, a MACSBUG™ type monitor along with an auto boot loader and a mnemonic disassembler, plus many more features.

All upgrades also come standard with REGULUS™, a UNIX like operating system which is totally compatible with UNIX. REGULUS supports real-time tasks, shared memory, record locking and contains a shell similar to the Berkeley C shell. Along with the operating system, you get C, an editor, assembler, linking loader, interactive debugger and a word processor.

SMOKE SIGNAL is also including in many of the kits the DCB-4A double density floppy controller which can handle up to four 5" and four 8" floppies and contains 1K of buffer RAM for fast disk transfers; the PSA-1 Winchester/Tape DMA interface board which has taps for SASI and Priam disk interfaces as well as a tap for 90 ips tape streamers which are supported under both REGULUS and OS9™; either a M-256-X or M-512-X dynamic RAM board with over two years of field proven reliability; and the LM8-1A heavy duty motherboard with gold plated connectors, extended addressing and on-board baud rate generator with ten selectable baud rates.

SOFTWARE

Software and Software Support is available only from Smoke Signal dealers. Spread Sheet, Word Processing, Relational Database, C, Basic and Cobol are all available now. Additional system's software is becoming available every day because of the UNIX compatibility.

SMOKE SIGNAL dealers are also offering Microware's OS9/68K™ to purchasers of these upgrade kits. SMOKE SIGNAL will offer other Microware 68000 products as they become available.

SUPPORT

Even at these "lower than PC" prices, we're not going to leave you with "PC" type support. We've arranged with one of our very technically qualified dealers to provide you with add-on software and technical support. In addition to answering your questions on how to convert your system to the 68000, he has a group of his customers who are themselves computer experts who are joining in a network that will help with even the most technical questions. We hope you will contribute your ideas to the network so that we all can benefit from new and fresh thinking. Complete details of the support available are included with the upgrade systems.

ORDER FORM

Fill in your name, address and phone number below. Your order will be shipped UPS so please do not use P.O. Box. Check items being ordered on form. Add prices for all items selected. CA residents must add 6% for sales tax. Total the amount for your order and check payment method below.

Name ______________________
Address ______________________
City, State, Zip ______________________
____
Phone ______________________

Payment: □ Enclosed Check
□ VISA
Mastercard

Card # ______________________
Exp. Date ______________________
Signature ______________________

□ SS-FD UPGRADE	$2100	________
□ SS-HD UPGRADE	500	________
□ SWTP UPGRADE	2800	________
□ GIMIX UPGRADE	2500	________
□ M-256X RAM	648	________
□ M-512-X RAM	948	________
□ SER-2 I/O	65	________
□ 20Mb HARD DISK (Instead of 5Mb)	800	________
Sub Total		________
CA residents add 6%		________
Total		________

SEND COMPLETED ORDER FORM TO:
SMOKE SIGNAL
31336 Via Colinas, Westlake Village, CA
91362-3984
(818) 889-9340

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Bells and Whistles or Bricks and Mortar?

I recently have been doing some thinking about software again. This topic is the result of my having looked at a couple of products over the past several months. As usual, let me say first that I am not out to pick on anyone in particular, so I will use no names. What I am saying is that the author of software normally becomes so involved in the detail that he can't see the overall picture very clearly. (I include myself in these remarks). Because of this tendency, sometimes, the software writer will include a "cute" feature of marginal use to the user, while completely overlooking some very frequently used feature, either omitting it, or making it very awkward to use.

The first example might be a screen editor that I received some time ago. I found that it would actually total values in a table so that totals could be run for rows or columns or both. That is a nice "whistle" by my definition. I've never needed such a function in an editor, and if I did, it would be so easy to use a calculator alongside to do the totals, and then type them in.... Now, I was going into this editor's features and I discovered that in order to indent a paragraph, I had to put the cursor at the first line of the paragraph and type a sequence of instructions. At that point, the paragraph would reformat. Now to indent the next paragraph, it was again necessary to bump the cursor down to that paragraph and type the sequence of instructions in again. Worse, any editing of the first line of any paragraph, would cause the indent to be lost, and the pr ess for that paragraph had to be repeated.

I imagine that most of you have used screen editors by now, and you know that most have some provision for indenting. The problem with the above mentioned editor could be fixed simply by modifying the paragraph formatter routine so it would ignore (and leave in place) any leading spaces at the start of a new paragraph. The spaces could be inserted by spacing or tabbing over a few spaces at the start of each paragraph, and would remain there permanently. When I wrote the supplier of the editor with the suggestion, he apparently didn't take kindly to my criticism, and he has not answered my letter. I don't know about everyone's applications for editors, but I would imagine that a paragraph indent would be performed vastly more times than totalizing the columns or rows in a table of numbers. I can only say that this editor was skimpy on the basic building blocks but contained unnecessary frills.

Example 2 is a compiler that I looked at recently. I found that it had a nice feature that would convert integer numbers to the equivalent Roman Numerals (in a choice of upper or lower case yet!). That is a nice feature that many users of the compiler would never use, and some would use infrequently. I found later that a couple of very standard features of the compiler had not been implemented. (In this case, the version I received was a "very preliminary" one, and the missing features are being implemented, but it struck me strange that the frills were done before the standard features).

When I wrote some software some time ago, I was guilty of doing some things "cleverly" that could have been done in a way much more standard to FLEX users. When this was pointed out to me by someone that received a copy for comments, I was defensive on first reading of the letter, but decided that the comments were quite valid, and I changed the software to make it operate in a manner much more as expected and in keeping with FLEX. I think the result was a greatly improved package.

Perhaps this indicates that not only is a software author too close to the project to be able to write the instruction manual, but that he may also be too close to the project to be objective about the features that are included. We all have software that is "almost". A second opinion would have made most of that software at least "great" if not perfect.

While I am throwing bricks, let me also pitch a couple of bouquets. Though there is reason to leave the bricks aimed at folks who will remain anonymous, there is no reason not to mention names for some positive feedback. Dan Farnsworth of Palm Beach Software (and of recent "great debate" fame) sent me preliminary versions of his SPELLB. I gave it a workout and responded with several suggestions, all of which were taken seriously. Both Dan and I feel that the final result was better for having had two people look at it.

When I first received PL/9 I made a number of suggestions for improvements in features, and I found Graham Trott, its author, most willing to listen. Many of the suggested improvements were implemented, and the rest at least brought forth an explanation of why they couldn't be added.

More Compiler Debate Evidence

I know, I said last time that the topic had been exhausted, but something has happened since then. Let me give you a case history of a project in which I have been involved. I've mentioned this previously, but it has come to a conclusion so I can report final results.

Several months ago, I received a call from a company that needed some help with software. They had a fairly complex calculation program that they had written for a 6800, in assembler. The execution time for the calculation was 160 seconds. "Send me the listing and a non-disclosure agreement, and I'll see what can be done", I said. The listing arrived, and I found that the math package was a BCD version. The scientific functions EXP, LOG, and XTOY (in BASIC X↑Y) were required. The functions were accurate enough, but with the slow BCD arithmetic and overly complex functions, XTOY took 5 seconds all by itself.

A look at the code for the calculations showed that the author had used B↑3 and X↑2 liberally rather than the much faster B*B*B and X*X. I changed those and then as a first step wrote a new SQRT function since the package used X↑0.5 for the square root, and I knew that to be a slow calculation. The result was an execution time of 16 seconds. Next step was to write more efficient scientific functions. That done, the time was down to 8 seconds.

I had mentioned switching to a binary math package, and we decided to go in that direction for further improvement. I had a math package that I had done previously, and I made some improvements to it, adding the necessary scientific functions. Now, of course the calculations had to be recoded to match the new math package. It was about a 12 hour chore to recode the 8 or ten pages of calculations and get them debugged. Resulting execution time was not quite what I had hoped, but it was approximately 2.75 seconds. I could probably, with some hours of looking and hair pulling manage to get that down to 2.5 seconds or a little less.

I wrote the customer that I felt this was about as well as we could do with a 6800. Because the 6809 has the MUL instruction that does an 8 bit by 8 bit multiply in 12 microseconds (six on a 2 MHz system) it should run considerably faster, particularly in light of the fact that the scientific functions use many multiplies and few divides, so as to take advantage of the '09's capabilities.

I have a 6809 version of the binary math package, but I would have to code the improvements to make it compatible. I decided to test my theory about the '09 advantage by coding the calculation in PL/9 and seeing how fast it would run. Would you believe that it ran in 0.280 seconds? The MUL instruction brought about a speed advantage of nearly ten times, in going to a 6809. I've still not coded the 6809 binary math package for compatibility so I can't yet test it, but I'm betting it won't be much better (if any) than the PL/9 results.

I draw several conclusions from all this. First, Assembler code is not inherently fast. It has to be efficiently coded to have the speed advantage so glibly claimed by its proponents. Second the pr essor DOES make a difference, though the advantage of one processor over another will depend greatly on the application. In a simple control application consisting mostly of AND and OR logic and shifts and rotates, the 6809 would have no speed advantage over the 6800, for example. Third, compiled code is not necessarily slow or inefficient. The PL/9 output was about 20% larger than the best assembler code version, and about the same size as the original assemb er code version.

As I stated above, the recoding of the calculations and the debug of the assembler version took about 12 hours. It took no more than half an hour to write the PL/9 code, and it was debugged in five minutes on the second try. Now someone convince me that assembler was a better choice for this application!! I think this was a fair test. The job to be done was clearly defined at the start. The chore was simply one of coding the calculations, and there was no advantage gained by having done one version ahead of the other. I am about as familiar with and experienced in assembler coding as in PL/9 so there was no advantage one way or the other due to familiarity. I wrote the scientific functions package for the assembler version, and was responsible at least in part for the package in PL/9, so that there was no familiarity bias in that area either. The fact is that it is much easier to code:

```
Z=(1-SQR(B*B+A*A*A)/3)↑(1/3)
```

Than it is to code:

```
JSR MATH
FCB PSH
FDB ONE
FCB PSH
FDB A
FCB CPY
FCB CPY
FCB FML
FCB FML
FCB PSH
FDB B
FCB CPY
FCB FML
FCB FAD
FCB PSH
FDB THREE
FCB FDV
FCB FSB
FCB PSH
FDB ONE
FCB PSH
FDB THREE
FCB FDV
FCB SWP
FCB EXT
JSR XTOY
JSR MATH
FCB STR
FDB Z
```

I made a couple of mistakes in coding the above example and had to correct them just now. Multiply this little example by 20 or so, and you get an idea of the order of magnitude of the project.

Maintenance Policy

I've recently been disturbed by what seemed to be restrictive policies on the part of Windrush Microsystems, and I wrote and told them so. Their reply to my criticism is well thought out and it explains their policies quite well, so, with their permission, I am going to explain here.

My irritation was over the fact that Windrush won't provide an update of a customer's compiler unless that customer returns the original disk on which the software was supplied, to England, fully insured. Windrush then will supply the update on the original disk, to the customer. My irritation was twofold. First, it takes a long time for a disk to make a round trip to England. Second, insurance and postage are not cheap. "Why can't you have U. S. distributors provide updates?" I asked. Well, I found out. Let me quote from Bill Dickinson's letter.

"The fact that we upgrade customers who report bugs free of charge has got to be worth the delay and cost of postage back to the U.K. If we wanted to get clever we would start charging maintenance like everybody else. I'll let you decide which of the(se) two policies is the better one:

1. You buy my product. You've got 90 days to report any problems with it. If you find a bug on the 91st day you have got to buy my maintenance contract to get the bug fixed. My maintenance contract costs $75.00 per year. You will have to send me the original disks to get bugs fixed on your maintenance contract. You've had my product for over a year. I just released a new version with all sorts of improvements. You have not been paying for maintenance so you will have to pay for the upgrade which is 50% of the purchase price of a new one. If you have paid me $75 a year for the past two years you will get the upgrade free.

2. You buy my product. If you find a bug I don't care if you bought it last week or two years ago, I will try to resolve the problem. If solving the problem means giving you the latest version of the product then you will get the latest version of the product free of charge. All you have to do is send me the original disk. You've had my product for over a year. I just released a new version with all sorts of improvements. All you have to do is send me $25.00 and the original disk. I will send you the latest version and a new manual (if applicable)."

Of course Windrush's guarantee is the latter. Bill's letter continues:

"Asking for the original disk back may seem to be a waste of everyone's time until you consider the following points:

"We do not keep any sales records from one year to the next. Therefore we have no way of verifying whether someone is a 'paid up' customer or a rip-off artist. If the customer can produce our original disk then he is considered to be legitimate. If he can't he isn't.

"It prevents the customer from selling his old copy to someone else, who then expects to get an upgrade for $25.00!

"We have a great desire to keep our administrative costs to a very low level. If our administrative costs start to rise then we will have to adopt policy #1 as it will be the only way we could recover the losses.

"We lose money on $25.00 upgrades and positively lose money when we give someone a copy of the latest version to fix a bug he has discovered in a product that is two years old. The least the customer should be willing to do is to send us his disk back at a cost of about $15.00 and wait bit for the replacement.

"It is not really practical to send 'upgrade' packs to the distributors to stock as there is always the danger that they will be sold and neither we nor the author see any money from the sale.

"The bottom line is that for us to be able to support policy #2 the customer has got to be willing to put up with a little delay. If you had the choice of paying us an additional $75.00 within 90 days of purchasing the product and $75 for every year of maintenance you wanted in return for the ability to phone us for a free copy of the latest version (without having to return the disk first) would you pay it? I rest my case."

Bill, thanks for the clarification and defense of your policies. Now they make rather good sense to me. I've always found Windrush responsive to reports of bugs. One thing else to remember is that it takes about ten days for a letter from the U.S. to make it to England and about the same the other way. Of course I am talking about air mail letters. In addition to the mail delay, software authors sometimes get busy or have other commitment so that they can't drop everything and search for a bug immediately.

Well, Don Williams tells me that I have been getting a little too long winded lately, so I am going to give him some space for some other stuff this time.

OS9 USER NOTES

by **Peter Dibble**
("**OS-9 Users Notes**" Columnist;
'68' Micro Journal)

The First Step Into OS-9

There has been some call recently for information for the beginning user of OS-9. Color Computer users new to OS-9 feel swamped by the number of details involved in the operating system. This column is an attempt to make OS-9 seem simpler to new users.

The OS-9 operating system has started to develope a reputation for complexity and obscurity -- in other words, user hostility. It is an unjust accusation. The thing that makes OS-9 appear confusing is the way it is presented. There are many subtle features in the operating system, and a large array of utilities. The manuals that come with it could help but don't. The OS-9 manuals were written as reference manuals, not tutorials. They drop everything on you at once. A new OS-9 user who is experienced with computers or very brave should read the manuals, wrap his mind around the whole thing, and sit down at the computer to enjoy OS-9. That is the quick, brute force, way to learn OS-9, but if it doesn't work for you, I recommend a gentler approach.

My copy of CoCo OS-9 includes about fifty commands. All these commands are important to at least some people, but most of them are only confusing to to new OS-9 users. The entire English language includes more than a hundred thousand words, but most people only use fewer than twenty thousand of them, and it is possible to communicate with a vocabulary of a thousand words or less. Operating systems like Unix and OS-9 are very much like English in that respect. Of all the commands available under OS-9 about a dozen are really necessary.

The bare minimum set of OS-9 commands are:
backup
copy
del
dir
edit
format
free
list
rename
shell The shell is the program which processes the commands you type into OS-9 and runs the other commands. Several commands are built into the shell. They are:
chd
chx
ex
w
kill
setpr The only shell commands that you really need to know are chd and chx, if you mean to do assembly language programming you will also need:
asm
debug If you will be using Basic09 you will need:
Basic09
RunB
GFX

Of all these commands there are four that need explanation especially badly. Format needs to be discussed because it is dangerous; if it is used carelessly it can destroy important information. BACKUP is a relatively fast way to copy an entire disk (it is a very good thing to get into the habit of doing this); perhaps a careful discussion of BACKUP will encourage people to use it more. Explaining DIR is a good excuse to say a few things about directories: one of the more important features of OS-9. CHX and CHD also relate to directories, and seem straightforward. What they are supposed to do is less important to a person with a OS-9 on a small computer than their unofficial side effects.

Format

The format command is the first one to use. Until a disk has been formatted it is unuseable to OS-9. The format command writes a pattern on the disk which marks the disk off into sectors (which amount to pigeon-holes for OS-9 to store data in). After writing the pattern format checks the disk to make certain the pattern is recorded correctly on the disk. If it isn't, format will note that the sectors where the errors occurred are faulty, and those sectors won't be used to store data. Format also writes some information which will be used to manage files on that disk. In the process of doing all this the format program completely erases the disk. If the disk is fresh out of a box of new disks you can feel certain that there is nothing on the disk that you care about, but, if it is one you are recycling, be very careful. After format is started any data that was on that disk is gone forever.

Put the disk you want to format in the drive you aren't using for the system disk (I'm going to assume you have your system disk in the drive OS-9 calls /D0, and the disk you want to format in drive /D1). Invoke the format command by typing FORMAT /D1 at the OS-9 prompt. The command line should look like:

OS9:FORMAT /D1 to which you should get the response:

```
COLOR COMPUTER FORMATTER
FORMATTING DRIVE  /D1
Y (YES) OR  N (NO)
```

READY? This is format giving you a chance to change your mind. It is also a way for you to format disks if you only have one drive, by asking format to format the disk in drive /D0 and replacing the system disk with the disk you want to format in at this point. In either case double check that you are about to format the correct disk. If you want to be especially safe take your system disk out of drive /D0 at this point even if you are formatting the disk in drive one. There is no danger at all of format writing on the wrong disk, but you can't be too careful. If you reply N to the READY? prompt format will quit immediately leaving the disk intact. If you reply Y, there will be a pause (23 seconds on my CoCo), then format will prompt you for a name for the disk. The prompt will look like:

DISK NAME: At this point enter the name you have assigned to the disk. The name can be up to 32 characters long and may include blanks. Follow the disk name with an ENTER. Format will now check the disk. As it checks each track on the disk it will write the track number to the screen in hexadecimal (base 16). If you have a thirty five track drive, the numbers will be from 000 to 022. Then format will print the message:

NUMBER OF GOOD SECTORS: $000276 If the number is smaller than 276 (a base 16 number which is 630 in decimal) some sectors were faulty.

If you want to demonstrate to yourself that format did something to the disk try the FREE command on the new disk. Enter the command FREE /D1. The command line should look like:

OS9:FREE /D1 The response should be something like:

```
disk name CREATED ON 84/01/24
CAPACITY: 630 SECTORS (1-SECTOR
CLUSTERS)
620 FREE SECTORS, LARGEST BLOCK
620 SECTORS
```

Where "disk name" in the first line of the response will be the name you gave the disk when you formatted it.

Backup

The next command to use after the format command is BACKUP. It is crucial to have a backup copy of each software distribution disk you have. If you make an error that damages the only disk with an important piece of software on it you will have to wait until you can get a replacement for the disk before you can use your computer again. Even if the time wasted waiting for the replacement disk isn't important to you, consider that replacement disks cost money.

Backup is a relatively fast way to create an exact copy of a disk. It has lots of options, but the simplest way to use the command is to just give the command BACKUP. The command line should look like:

```
OS9:BACKUP
```

The response will be:

```
READY TO BACKUP FROM /D0 TO /D1
?:
```

At this point put the disk you want to copy in /D0 and a formatted disk which has nothing you want to keep on it in drive /D1. Then check the disk in /D1 ... BACKUP will erase anything that's on that disk. When you are certain everything is OK type Y. Now BACKUP will double check with you by telling you the name of the disk in drive /D1. The message will look like:

```
THE DISK
 IS BEING SCRATCHED
OK ?:
```

if you reply Y to this, the backup from the disk in /D0 to the disk in /D1 will take place. The disk in /D1 will become an exact copy of the disk in /D0 right down to the disk's name.

The BACKUP command takes what seems like a long time to run. There are two things that can speed it up. One is to use the -V option which prevents the copy from being verified. I don't suggest that anyone use this option. The other way to speed BACKUP up is to instruct OS-9 to give it extra memory to run in. BACKUP can use extra memory to run more quickly. BACKUP ran for one minute 58 seconds when I started it with the command line:

```
OS9:BACKUP
```

Normally BACKUP uses 19 pages of memory. If you give it more -- say 100 pages -- with the command line:

```
OS9:BACKUP   100
```

it runs in one minute 48 seconds. It is also quieter because the heads on the disks don't load and unload as often.

Dir

The command which tells you what files are one your disks is the the Dir (short for directory) command. If you just type DIR after booting OS-9 you will get a response like

```
DIRECTORY OF .  23:55:08
OS9BOOT    CMDS      SYS
DEFS       STARTUP
```

This means that you are listing the current directory which is known by the pseudonym "." at 11:55:08 in the evening. The files in that directory are OS9BOOT, CMDS, SYS, DEFS, and STARTUP. Now, in fact only OS9BOOT and STARTUP are normal files, the other three files are subdirectories. Subdirectories are such an interesting topic that they were the subject of their own column some months ago, and won't be covered any more than absolutely necessary here. To find out more about the files than their names use the command DIR E.

```
     OS9:DIR E which will respond:
      DIRECTORY OF .  23:59:57

     CREATED ON   OWNER    NAME
       ATTR     START       SIZE
     ==============================
     83/06/02 1921      0   OS9BOOT
     ------WR          A      3032
     83/06/02 1956      0   CMDS
     D-EWREWR         3C       6A0
     83/06/02 2002      0   SYS
     D-EWREWR        164        A0
     83/06/02 2002      0   DEFS
     D-EWREWR        17F        C0
     83/06/02 2003      0   STARTUP
     ----R-WR        1F5         E
```

then it will stop because the screen is full. When you are ready to continue hit any key ... I usually press the space bar. That was the end of the directory, so all you get after you let the output continue is a few blank lines and a new OS9 prompt.

Two of the fields in the DIR E output are of no special interest until you become an advanced OS-9 user: OWNER, and START. The first two fields for each file are the date and time the file was created. The date is in the usual YY/MM/DD format and the time is in HHMM format with hours ranging from 00 to 23. The attributes field contains information about what the file can be used for. The important thing right now is that files with a D as the first character in the attribute field are directories. Files with a dash as the first character in their attribute field are normal files.

The other option which can be used with the DIR command is X. The X option is a short hand way to get the directory of the execution directory; that is, the directory OS-9 searches for programs, like the commands, you ask it to run. The command line:
     DIR X will give you a rather long list of all the files in your execution directory. If you haven't written any of your own programs, this will be a list of all the commands and utility programs which came with OS-9. You will probably have to press the space bar in the middle of the output of this command. It is more than one page long.

Chx and Chd

Chx stands for Change Execution Directory, Chd for Change Data Directory. OS-9 expects to find all commands, whether they are part of the operating system or something you wrote, in the execution directory. All files that you don't mean to execute are looked for in the data directory. (There are ways around both of these restrictions, but let's skip that for now.) After you boot OS-9 you will find that the execution directory is /D0/CMDS and the data directory is /D0. If you have a second drive (I have been assuming that you do) you will probably want to use that for data. The command:
     CHD /D1 will cause all future references to data files to look for them on /D1.

To speed OS-9 up, the location of the directory file on the disk is kept in memory. This leads to the important side effect of the Chd and Chx commands. When you read the directory OS-9 goes directly to the directory's location on disk and starts reading ... imagine what would happen if you fooled OS-9 by changing disks. You change disks and type a command like
     LIST FOO or even just DIR. Your operating system will start reading where the directory is supposed to be. Since the disk with a directory at the selected spot is sitting in its envelope and some other disk is in the drive, OS-9 will find something unexpected where the directory was. The result could be any of a number of error messages. The solution to this problem is to always give OS-9 a chance to locate the directories on a new disk by giving it Chd and Chx commands as necessary when you change disks.

There is one last tricky thing about the Chx/Chd commands' special use. If you keep things very simple it will seem that you only need to use the Chx command, but this is just a special case. I suggest that you learn how to make directories and use them as soon as you can, but, until you start using them, the new disks you use to store data will only have the directory FORMAT automatically creates (called the "root directory"). The root directory is always at the same location on a disk. Because of this special fact about the root directory OS-9 is always able to find it, and changing disks that only have the root directory on them won't cause any trouble. The execution directory is usually not the root directory, so this special case doesn't generally apply to it.

The set of commands I have mentioned in this column might be considered a "starter set" for OS-9. The dozens of commands I left out are certainly worth learning, but you can get OS-9 working with these few.

Oops

I neglected to mention a few months ago that OFlex as reviewed in this column is available only from Gimix. Richard Don, the salesman for Gimix, explained the geneology of OFlex to me. It is Flex by TSC adapted by Richard Hogg to run under OS-9. Gimix provides enhanced disk Device Drivers to support Flex's requirements, and made some enhancements to Richard Hogg's design. Anyone who takes out licenses from TSC and Richard Hogg can sell OFlex, but the version I reviewed has features added by Gimix.

"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, GA 30207

INTRODUCTION

This month's column discusses the new Tandy C compiler for COCO OS/9 and provides an example of the use of the C language in interfacing with interrupt-driven devices: in particular, the COCO printer port.

TANDY C COMPILER

Tandy has recently begun marketing the Microware (McCosh) C compiler for the COCO for $99.00. This is in sharp contrast to the Microware cost of $250.00 for essentially the same product. It includes its own relocating macro assembler, for which Microware asks $125.00, if purchased separately from the C compiler.

For a COCO OS/9 user, the situation is excellent, in that Tandy has made a full C compiler available for $99.00, along with reasonably good documentation, which is a typeset version of the Microware C manual. Even the relocating macro assembler itself may be worth the $99.00.

For non-COCO OS/9 users, the situation is not quite so clear. Although the price is very good, there are several non-monetary disincentives to the use of the Tandy C.

One obstacle which must be cleared is the COCO OS/9 format of the Tandy C diskette. There are several means of solving this problem. If a COCO is available, the D. P. Johnson SDISK software may be used to copy it to a standard OS/9 mini-floppy format. If no COCO is available, but a GIMIX system with mini-floppies is available, the GIMIX COCO driver may be used to read the diskettes. If mini-floppy diskettes are not acceptable, the files may be transmitted with one of the available modem programs from one system to another.

A small problem with the Tandy C package concerns the fact that it does not include the K & R C book, while the Microware C package contains it. The highest price noted for this book is $19.95, from Microware.

Another disadvantage with the Tandy C for OS/9 Level 2 users is that only the version of C for OS/9 Level 1 is included, whereas the Microware C provides versions of C for both OS/9 Level 1 and OS/9 Level 2. There may be subtle differences in the versions of the compilers for the two levels of OS/9, but I am aware of no differences aside from the obvious one of two passes for the OS/9 Level 1 version.

For new users, the Tandy C compiler is supported by Tandy service and the Microware C compiler is supported by the Microware hot-line. Both services are free for the first ninety days, and available for a fee subsequently. The Microware support costs $150.00 per year for all products.

Following is a table representing the combined alphabetized directories of each of the C compiler release diskettes.

Microware/McCosh	Tandy
cmds.common/c.asm	cmds/c.asm
cmds.common/c.link	cmds/c.link
cmds.common/c.opt	cmds/c.opt
cmds.common/c.prep	cmds/c.prep
cmds.II/c.com	
cmds.II/c.pass1	cmds/c.pass1
cmds.II/c.pass2	cmds/c.pass2
cmds.II/cc1	cmds/cc1
cmds.III/c.comp	
cmds.III/cc2	
defs/ctype.h	defs/ctype.h
defs/direct.h	defs/direct.h
defs/errno.h	defs/errno.h
defs/modes.h	defs/modes.h
defs/module.h	defs/module.h
defs/os9.h	defs/os9.h
defs/os9defs.a	defs/os9defs.a
defs/setjmp.h	defs/setjmp.h
defs/sgstat.h	defs/sgstat.h
defs/signal.h	defs/signal.h
defs/stdio.h	defs/stdio.h
defs/time.h	defs/time.h
lib/clib.l	lib/clib.l
lib/cstart.r	lib/cstart.r
sources/line.c	sources/line.c
sources/prof.c	sources/prof.c
sources/rdump.c	sources/rdump.c
sources/sys/abort.a	sources/sys/abort.a
sources/sys/access.a	sources/sys/access.a
sources/sys/ccdevice.a	
sources/sys/cfinish.a	sources/sys/cfinish.a
sources/sys/change.a	sources/sys/change.a
sources/sys/comp.sys	sources/sys/comp.sys
sources/sys/cstart.a	sources/sys/cstart.a
sources/sys/dir.a	sources/sys/dir.a
sources/sys/id.a	sources/sys/id.a
sources/sys/intercept.a	sources/sys/intercept.a
sources/sys/io.a	sources/sys/io.a
sources/sys/make.sys	sources/sys/make.sys
sources/sys/mem.a	sources/sys/mem.a
sources/sys/misc.a	sources/sys/misc.a
sources/sys/mod.a	sources/sys/mod.a
sources/sys/process.a	sources/sys/process.a
sources/sys/profdummy.a	sources/sys/profdummy.a
sources/sys/signal.a	sources/sys/signal.a
sources/sys/stat.a	sources/sys/stat.a
sources/sys/syscall.a	sources/sys/syscall.a
sources/sys/syscommon.a	sources/sys/syscommon.a
sources/sys/tidyup.a	sources/sys/tidyup.a
sources/sys/time.a	sources/sys/time.a

The only differences between the versions, in terms of the file structures of the release diskettes, involve the organization of the compilers and the file "sources/sys/ccdevice.a". As far as could be determined, the contents of the corresponding non-executable files on the release diskettes are identical. The "ccdevice.a" file provides a simple means of changing the assumed device containing the DEFS and LIB files. Since this is assumed not to be a problem with COCO OS/9, it is not provided on the diskette. However, the device strings in the compiler and preprocessor could be found and patched if this were a problem in a particular environment.

The objection to the use of the Tandy version of C on non-Tandy systems may present a problem to copyright purists. Copyright lawyers disagree on such usage, but most do not envision a conflict with the existing copyright laws, assuming the product is not used for resale. The standard Tandy copyright notice in the manual does not prohibit the use of the software on a non-Tandy system, and allows the making of backup copies for the customer's use.

Hopefully, this discussion has clarified, not obscured, the issue of the use of Tandy C on non-COCO systems.

INTERFACING C PROGRAMS WITH INTERRUPT-DRIVEN DEVICES

Recently, I had been attempting to get a modem program working properly under COCO OS/9, using the internal printer port. The program is written in C and was already working on several non-COCO 6809-based computers.

The primary problems involved the original RS232 driver module, which had several serious errors, including the lack of a status function and its inability to process characters properly at any rate over 300 Baud. After procuring a replacement module from Dale Puckett and rewriting it to correct the status function and other problems, its performance continued to be unacceptable.

The primary problem with the original and revised RS232 modules concerns the bit-banging nature of the printer port when used for input. The original module polls for characters on each interrupt and has no input or output character queues. The revised module uses the FIRQ interrupt to signal the beginning of an input character, and places the characters into an input queue, but normally receives characters properly only until its queue is full.

With either module, to ensure proper timing, interrupts must be masked during the reception or transmission of characters from or to the printer port. The input routine waits until half of the first bit time has elapsed, then checks the status of the line for each bit time to determine which bit pattern is on the input line. The output routine places the appropriate signal on the output line for the time corresponding to each bit time, in order to construct the ASCII character code, with transmission envelope of one start bit and one stop bit.

All other processing on the COCO is restricted to the time between character reception or transmission. If characters are being received or transmitted continuously, the time between characters is only about ten percent of the available time for processing. The overhead involved with OS/9 uses up most of this small amount, leaving the program an insufficient amount of time to process input data, even if only stuffing the characters into memory, and the queue overflows if enough characters are received to fill it.

I decided to change the concept of the character collection process to that of an interrupt-driven handler depositing the incoming characters into a large (possibly circular) buffer, from which the C program could draw the characters. This helps reduce the OS/9 overhead and postpones the queue-full situation until the circular buffer overflows, which may not occur if the user is careful in what the COCO is requested to receive.

The real point of the inclusion of the C functions described below is not to demonstrate how to use the printer port on the COCO, but to provide a concrete example of the use of C programs with interrupt drivers. Many other situations will be similar to this one, and will often be more complex, especially in the area of output.

The COCO printer connector must be wired as follows:

pins 1 and 2	received data
pin 3	ground
pin 4	transmitted data

In order to allow a FIRO interrupt to be generated at the beginning of each character.

The output characters are written directly to the port, since there is no interrupt to indicate that the port is ready for output, as in the ACIA and other chips more suited for serial output. Characters are transmitted and received as eight-bit bytes with no parity.

The C program controls the processing of the input and output characters thru a group of functions, the logic of which is implemented thru assembler-language text. They are as follows:

xinitpia	sets up pia and firq
xtermpia	resets pia and firq
xresbuf	resets buffer pointers
xmaskpia	disables firq
xunmkpia	enables firq
xoutpia	transmits one character

Xinitpia establishes the boundaries of the large buffer, the addresses of the head and tail queue pointers, the address of the overflow flag, the circularity state of the buffer, and the baud rate delay factor. It establishes the firq vector, but disables the firq interrupt itself. It calls xresbuf to reset the queue pointers and overflow flag.

Xtermpia restores the original firq vector and disables the firq interrupt.

Xresbuf resets the head and tail queue pointers to the beginning of the buffer, sets the circularity state of the buffer, and resets the overflow flag.

Xmaskpia disables the firq interrupt.

Xunmkpia enables the firq interrupt.

Xoutpia transmits a character at the specified baud rate.

Although it may bother OS/9 purists, the functions share a work area into which they store data on a pc-relative basis. This could have not easily been avoided, because of the interrupt-driven nature of the code. However, the COCO has only one printer port, and thus it can logically have only one owner active at any point in time, so sharable code is not a problem.

There are a few rules the C driver program must obey. The primary one is that it may not arbitrarily manipulate the queue head and tail pointers. In particular, it should not directly modify the head pointer and should advance the tail pointer by one and only after removing each character from the buffer. If the buffer is circular, the tail pointer must be reset to the beginning of the buffer when it overflows the end of the buffer. The overflow flag is actually a counter of the number of characters which could not be placed into the buffer; it may be zeroed after interrogation.

The COCO must not be requested by any user, not only the printer port user, to perform operations (such as disk I/O) which would cause interrupts to be masked during periods of character reception, or input characters may be lost. Since OS/9 itself periodically masks interrupts to perform such operations such as timer service routines, the exact time delay between the start of the character and the start of the firq handling routine will vary slightly. This may cause an occasional garbled input character at higher baud rates.

The program must call xinitpia to establish the routine's necessary pointers, vectors, and flags. During periods when no characters are to be placed into the buffer, the xmaskpia function should be used. Conversely, when characters are to be allowed into the buffer, the xunmkpia function should be used. The xtermpia function must be used before the program terminates; it disables the firq interrupt and restores the firq vector.

The baud rate delay factor specifies the number of delay loops required to correctly receive or transmit characters on the printer port. The following table provides the suggested delay factors for certain standard baud rates:

baud rate	delay factor
110	$04cc
300	$01bf
600	$00de
1200	$006d
2400	$0036
4800	$001b
9600	$000e
19200	$0007

Because baud rates may vary by as much as ten percent from device to device, some modification in the suggested values may be required in specific cases to reduce the error rate.

The functions just described are presented below, along with a simple main function which displays the characters input into the buffer from the printer port. In order to provide a termination point, the buffer is made non-circular and the program terminates when the buffer is full.

```
main()
{
    char buff[1000];
    char *head,*tail;
    int over;
    xinitpia(&buff[0],&buff[999],&head,&tail,&over,0,0x6e);
    xunmkpia();
    while (tail<=&buff[999])
    {
        while (head==tail);
        putchar (*tail++);
    }
    xtermpia();
}

xinitpia (buff,bufe,head,tail,over,circ,baud)
char *buff,*bufe,*head,*tail,*over;
int circ,baud;
{
#asm
****************
* initpia (&buff,&bufe,&head,&tail,&over,circ,baud)
*   initialize pia and start processing
*
initpia pshs cc,dp,x
 orcc #$50 mask interrupts
 ldx #$ff20 point to pia
 clr $01,x data direction
 lda #$fe set rs-232 lines
 sta ,x mostly output
 lda #$36 set data register
 sta $01,x
 lda #(parlas-parfir) initialize internal storage
 leax $0a,s point to first parameter in stack
 leay parfir,pcr point to storage
initlp ldb ,x+ copy parameter list
 stb ,y+
 deca
 bne initlp
 ldd circ,pcr reset buffer pointers and flags
 pshs d,x,y,u
 lbsr resbuf
 puls d,x,y,u
 leax basicp,pcr set new firq vector
 ldy #frqos9 os/9 firq vector address
 ldd #$7ecd check for flex
 cmpd $cd00
 bne initos
 cmpd $cd03
 bne initos
 ldy #frqflx flex firq vector address
 ldb ,y flex firq jump
 stb oldjmp,pcr
 sta ,y+ jump
initos sty oldadr,pcr
 ldy ,y remember firq vector
 sty oldfrq,pcr
 stx [oldadr,pcr] replace firq vector
 puls cc,dp,x
 lbra overfirq
***************
```

```
* basicp
*   basic pia input driver
*
basicp pshs d,dp,x,y,u stack other registers
 leax worka,pcr build dp-register
 tfr x,d
 tfr a,dp
 ldd baud,pcr baud delay factor
 cmpd #$000b >9600 baud
 bhi basic1 wait for rest of start bit
 ldb #$08 get bit count
 pshs b hold it
 bra basic5 get the character
basic1 lsra divide delay factor in half
 rorb for start bit
 subd #$0005 overhead
basic2 subd #$0001
 bne basic2 time half bit
 ldb #$08 get bit count
 pshs b hold it
basic3 ldd baud,pcr baud delay factor
 subd #$0005 overhead
basic4 subd #$0001
 bne basic4 time full bit
basic5 ldb $ff22 get incoming bit
 lsrb hold bit in cc
 ror <worka build the character
 dec ,s decrease bit count
 bne basic3 done with character?
 puls b
 lda $ff20 clear pending interrupt
 lda <worka get character
 ldx [head,pcr] get head pointer
 sta ,x+ put character into buffer
 cmpx [tail,pcr] check tail pointer
 beq basic7
 cmpx bufe,pcr check for end of buffer
 bls basic6
 ldd [circ,pcr] check circular flag
 beq basic7
 ldx buff,pcr reset to start of buffer
basic6 stx [head,pcr] set new head pointer
 bra basic9
basic7 ldd [over,pcr] incr overflow flag
 addd #$0001
 beq basic9 ensure no rollover
 std [over,pcr]
basic9 puls d,dp,x,y,u restore other registers
 rti return to the original task
***************
* interrupt handler storage areas
*
parfir equ * first parameter
buff fdb $0000 address of start of buffer
bufe fdb $0000 address of end of buffer
head fdb $0000 address of head pointer
tail fdb $0000 address of tail pointer
over fdb $0000 address of overflow flag
circ fdb $0000 circular buffer flag
baud fdb $0000 baud rate delay factor
```

```
parlas equ * last parameter
oldadr fdb $0000 old firq address
oldjmp fcb $00 old firq jump
oldfrq fdb $0000 old firq vector
worka fcb $00 work area
frqos9 equ $0030 os/9 firq vector
frqflx equ $010f flex firq vector
overfirq equ *
#endasm
;
}

xtermpia ()
{
#asm
***************
* termpia ()
*   terminate pia processing
*
termpia pshs cc,x
 orcc #$50 mask interrupts
 bsr maskpia mask firq from pia
 ldd oldfrq,pcr restore firq vector
 ldx oldadr,pcr
 std ,x
 cmpx #(frqflx+1) check for flex
 bne termpix
 lda oldjmp,pcr restore jump
 sta -$01,x
termpix clra return zero
 clrb
 puls cc,x
#endasm
;
}

xresbuf (circ)
int circ;
{
#asm
***************
* resbuf (circ)
*   reset buffer pointers and flags
*
resbuf pshs cc
 orcc #$50 mask interrupts
 bsr maskpia mask firq from pia
 ldd buff,pcr
 std [head,pcr]
 std [tail,pcr]
 ldd $07,s
 std circ,pcr
 clra
 clrb
 std [over,pcr]
 puls cc
#endasm
;
}
```

```
xmaskpia ()
{
#asm
***************
* maskpia ()
*   disable firq from pia
*
maskpia pshs cc
 orcc #$50 mask interrupts
 lda #$34 disable firq from pia
 sta $ff21
 lda $ff20 clear pending interrupts
 clra return zero
 clrb
 puls cc
#endasm
;
}

xunmkpia ()
{
#asm
***************
* unmkpia ()
*   enable firq from pia
*
unmkpia pshs cc
 orcc #$50 mask interrupts
 lda #$35 enable firq from pia
 sta $ff21
 lda $ff20 clear pending interrupts
 clra return zero
 clrb
 puls cc
#endasm
;
}

xoutpia (chr)
char chr;
{
#asm
***************
* outpia (chr)
*   send one character to pia
*
outpia ldb #$09 number of bits to output
 pshs b,cc store bit count and interrupt flags
 clrb clear carry for start bit
 orcc #$50 mask interrupts
outpi1 ldb #$02 pia mask for 1 bit
 bcs outpi2 if carry set, write a 1 else write 0
 clrb pia mask for 0 bit
outpi2 stb $ff20 put the bit on the line
 ldd baud,pcr baud delay factor
 subd #$0006 overhead
outpi3 subd #$0001
 bne outpi3 time full bit
 lsr $09,s shift character for next bit
 dec $01,s count bits
 bne outpil continue if more bits
 ldb #$02 pia mask for 1 bit
 stb $ff20 put stop bit on line
 ldb $ff20 clear interrupts
 puls cc,b
 clra return zero
 clrb
#endasm
;
}
```

SUMMARY

This month's column discussed the Tandy C compiler for COCO OS/9 and the use of a C program in driving the COCO printer port using interrupts to indicate input operations in order to attempt to receive the characters more accurately.

68000 USER NOTES

Philip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

For this month's column, I have the results of the prime number benchmark on my 68008. There are also listings of the C and assembler programs used in the test. I also will cover the various programs included in the OS-9/68K package a little more deeply.

A Speed Test

What would a new computer be without someone immediately testing its speed? To perform the test, I used the prime number Sieve of Eratosthenes program, which has appeared in Ron Anderson's column, as well as in Byte magazine articles on benchmarking. The test consisted of various versions of some C programs, as well as two 68000 assembler programs. All were run on a 10 MHz 68008, and the C programs were also run on a 2 MHz 6809 for comparison. Microware's C compiler was used with both the 6809 and the 68008. The programs were run for 100 iterations, with the times in the table adjusted for 10 iterations for comparison with the previously published benchmarks.

The first C program is a direct copy of the program found in the January 1983 Byte, page 284. I first ran the program as written, with the scalar variables all auto variables, that is, variables located on the local stack. I then changed the variable declaration from 'int' to 'register int'. For the 6809, this caused the variable 'i' to be kept in the U register. For the 68000, though, this register declaration caused all of the variables except 'iter' to be kept in registers D4 to D7. I then wrote an assembler program which closely mimicked the algorithm of the C program.

The first C and assembler programs refer to the 'flags' array using indexing. That is, references in C are of the form 'flags[i]', and references in assembler are of the form '0(a0,d0)' where A0 holds the address of 'flags[0]', and D0 holds the value of the variable 'i'. While keeping with the same general algorithm, I wrote improved versions of both programs, in which the array is referenced using pointer variables, which hold the address of the current entry. The C program was then run with auto variables and register variables. For the 68000, all seven of the variables in this version were kept in registers, using D4 to D7 and A2 to A4.

Note that, for the C versions using auto variables, the 6809 was about as fast as the 68008. When register variables were used, though, the 68008 really flew. In fact, the C times, then, were not all that different from the times for the assembler code. The speed of utilities and the like written in C should prove to be quite good, as long as the large register set in the 68000 is properly utilized. I'm not sure what the times would be for a true 16 bit 68000, but my guess would be about 60% to 75% of the times for the 68008.

So what does the test show? In various tests of 6809 compilers, high level language versions generally take about 10 seconds at 2 MHz, and my times are no exception. When register variables are used on the 68008, though, the times are reduced to less than half what they were. As far as assembler goes, the second, pointer version of the program is a better comparison for the 6809 assembler test in the August 1984 68MJ. The 2 MHz time there was 3.3 seconds, while here it is 2.2 seconds.

There is another point that can be made here. To get really good times for the 68008 using a high level language, I needed to be able to keep variables in the registers at all times. As far as I know, C is the only commonly available language with this ability. Without it, a 68000 would offer little significant improvement in speed over a 6809. It seems, then, that as our microprocessors evolve, the languages must do likewise. I'm not really suggesting that Pascal or the like should allow register declarations. We do need to have languages that do a much better job of optimizing the object code that they produce, to take advantage of the power inherent in the new crop of chips, though.

More on OS-9/68K

I'm getting this column out a little earlier than I had planned, so I haven't recieved version 0.6 of OS-9/68K yet. The remarks I have to make apply to version 0.5. I'll have more to say as the new versions come in. In any case, these are all still preliminary releases.

First of all, most of the utilities that I recieved with Level 2 OS-9/6809 are present with the 68000 version. Missing are display, echo, ident, verify, binex, and exbin. I don't particularly need binex and exbin, but the others come in handy. In fact, I wrote my own display utility since I need it to configure my terminal in the startup file. Hopefully, these should all be present in later versions.

Most of the utilities run a little differently here. The biggest difference has to do with the handling of options in the command line. Now, all options must be preceded with a minus sign. This may be the hardest adjustment to make when moving from OS-9/6809. I keep typing 'dir e' when I should type 'dir -e'. Most utilities now allow an option of '-?', which causes a summary of the command to be printed on the standard error output.

As I mentioned last month, the dir command has been made more powerful, with the addition of wildcard searching, as well as an option to output the files found one per line, without any header information. Most utilities which accept a number of filenames in the command line also accept the option '-z', which allows the list of files to come from a text file or the standard input. Piping the results of a wildcard dir command into one of these utilities gives the equivalent of the powerful wildcard file specification found in Unix, which comes in very handy when performing such jobs as deleting all of a group of related files.

There are several new utilities included with the package. The 'inlz' utility performs an I$Attach command on memory modules. This command can be used to perform the initialization of a device driver. I have used it to start up a RAM disk drive which is part of OS-9/68K. By running 'inlz r0', 32K of memory is set aside as a fast disk drive, referred to as /r0.

A utility by the name of 'cfp', for Command File Processor, provides a method for submitting procedure files with limited parameter substitution. A text file is created on disk, with an "*" in place of a filename. Cfp will read this file, and substitute the filename supplied on the command line for each occurence of the "*". The resulting commands are stored in a procedure file and executed.

There is a 'debug' program, which is quite a bit more powerful than the 6809 version. The main reason for this is the ability of the 68000 to automatically trace a program, one instruction at a time, as it is run. OS-9/68K provides all the links required to use this, and debug does a good job of it. Debug cen disassemble a portion of a program, and can also step through a program, disassembling as it runs. This was available in the 6809 with the Flex Debug, and was missed (by me at least) under OS-9/6809.

There are two editors included with OS-9/68K. The first is a line editor named 'edt'. Edt is useful with small text files. It functions in much the same way as the editor found in Basic09 (without the Basic language syntax checking, of course). The other editor is a screen editor, 'scred'. Scred combines a command mode, for doing such jobs as global searches and changes, with edit and insert modes, which operate with automatic screen updating. In my release, scred required some customizing before it could be run. The main part of the program was in a relocatable file 'scred.r'. A C source code file, 'sconfig.c', contained all of the terminal-specific routines, and I had to modify it to work with my equipment, and then compile and link the result to get a working program. Unfortunately, this requires the C compiler to work. Future versions may come with a number of pre-configured 'sconfig.r' files, requiring you to link in the one which applies to your own terminal. This would eliminate the need for the C compiler, which does not come as part of the OS-9/68K package.

The standard assembler which comes with OS-9/68K is a relocatable assembler, 'r68', which performs much like the 6809 assembler 'rma'. There is no absolute assembler like the 6809 'asm'. There is also a linker, 'l68', for converting relocatable code into executable modules. Read through the two assembler listings that were used in the primes benchmark to see an example of the source code accepted by r68. Notice that the second program is meant to be used in conjunction with the C library. The program calls the 'printf' routine to display some results at the completion of the run.

I did not recieve any manuals for r68 or l68. This was not much problem, but I did discover something having to do with the data area which is not immediately apparent. Look at the first assembly language program, which does not use the C library. The program starts by adding $8000 (32K) to register A6. In the 68000, indexing is within a 64K range, using a signed 16 bit value. That means that locations up to 32K forward or backwards of an address register's contents can be referenced. To address a full 64K in the data area, the address register must point 32K past the start of the data area. You might think that this is not required with my particular program, since the data area is well under 32K long. It turns out, though, that the linker assumes that a reference to a data area label is made with an address register that has been offset. The label 'flags', for instance, is assigned a value of -32768 (-$8000) by the linker, instead of the expected value of 0. Since a module running under OS-9/68K starts up with A6 pointing to the start of the data area, I had to add 32K to A6 to address the correct location using data area labels. I did not need to perform the offsetting in the second program, since this is done within the 'cstart' routine which actually begins all programs produced with the C compiler.

I'm Out of Room!

There are some long listings with this column, so I better cut it short here. Look here next month for some thoughts on operating systems, and what I might like to see. I just bought a book, 'The Unix Programming Environment', which was written in part by Brian Kernighan, who co-wrote the standard book for the C language (he's the K of K&R). The book describes how to use the Unix system, and spends a lot of time on using the shell. Just a quick glance makes me wish that some of the shell capabilities were included in OS-9/68K, and since memory usage is of less concern with the 68000, I see no reason that they could not be implemented. Anyway, more on this next month.

```
Sieve benchmark timings - 10 iterations - in seconds

        C - auto   C - register  Assembler

Version 1 - flags[] referenced with indexing

6809      11.0         9.7          ---
68008      9.7         4.8          3.5

Version 2 - flags[] referenced with pointers

6809       8.9         8.2          ---
68008      9.4         3.0          2.2
```

```
/* Erotosthenes Sieve Prime Number Program in C */

#define TRUE    1
#define FALSE   0
#define SIZE    8190
#define MAXLOOP 100

char flags[SIZE+1];

main()
{
    int i,prime,k,count,iter;

    printf("%d iterations.\n",MAXLOOP);
    for (iter = 1; iter <= MAXLOOP; iter++) {
        count = 0;
        for (i = 0; i <= SIZE; i++)
            flags[i] = TRUE;
        for (i = 0; i <= SIZE; i++) {
            if (flags[i]) {
                prime = i + i + 3;
                for (k = i + prime; k <= SIZE; k += prime)
                flags[k] = FALSE;
                count++;
            }
        }
    }
    printf("%d %d\n",prime,count);
}
```

```
/* Erotosthenes Sieve Prime Number Program in C */
/* Version 2 - use pointers for flags array reference */

#define TRUE    1
#define FALSE   0
#define SIZE    8190
#define MAXLOOP 100

char flags[SIZE+1];

main()
{
    register int i,prime,count,iter;
    register char *flagsi, *flagsk, *flagsend;

    printf("%d iterations.\n",MAXLOOP);
    flagsend = flags + SIZE;
    for (iter = 1; iter <= MAXLOOP; iter++) {
        count = 0;
        for (flagsi = flags; flagsi <= flagsend; )
            *flagsi++ = TRUE;
        flagsi = flags;
        for (i = 0; i <= SIZE; i++) {
            if (*flagsi++) {
                prime = i + i + 3;
                for (flagsk = flagsi + prime - 1; flagsk <= flagsend;
                      flagsk += prime)
                    *flagsk = FALSE;
                count++;
            }
        }
    }
    printf("%d %d\n",prime,count);
}
```

```
                         nam     sieve1
                         ttl     primes         by sieve benchmark

* sieve
*
* benchmark, in 68000 assembler
* finds primes using sieve of Eratosthenes

                         opt     n
                         opt     f

                         use     ../defs/oskdefs

 00000101 Type           set     (Prgrm<<8)+Objct
 00000001 Revs           set     (ReEnt<<8)+1
 00000001 Ed             set     1              edition
 00000200 Stk            set     512            stack allocation

                         psect   Sieve,Type,Revs,Ed,Stk,Entry

 00001ffe size           equ     8190           flags array size - 1
 00000064 maxiter        equ     100            number of iterations

                         vsect
 00000000 flags          ds.b    size+1
 00000000                ends

* Register usage - Variable names are from 'C' program
*   D0.w = i     - main index in flags array
*   D1.w = prime - current prime number
*   D2.w = k     - cross-out index in flags array
*   D3.w = count - count of primes found
*   D4.w = iter   - main iteration loop counter
*   A0.l = pointer to start of flags array

0000 ddfc Entry      adda.l   #$8000,a6       offset data space pointer
0006 41ee            lea      flags(a6),a0    point to flags array
000a 7863            moveq    #maxiter-1,d4
* count = 0
000c 4243 iterloop   clr      d3
* for (i = size; i >= 0; i--)
*     flags[i] = 1;
000e 303c            move     #size,d0
0012 11bc fillloop   move.b   #1,0(a0,d0.w)
0018 51c8            dbra     d0,fillloop
* for (i = 0; i <= size; i++)

001c 4240            clr      d0
001e 0c40 mainloop   cmpi     #size,d0
0022 6224            bhi.s    done
*      if (flags[i]) (
0024 4a30            tst.b    0(a0,d0.w)
0028 671a            beq.s    loopend
*          prime = i + i + 3;
002a 3200            move     d0,d1
002c d241            add      d1,d1
002e 5641            addq     #3,d1
*          for (k = i + prime; k <= size; k += prime)
*               flags[k] = 0;
0030 3400            move     d0,d2
0032 d441            add      d1,d2
0034 0c42 crossout   cmpi     #size,d2
0038 6208            bhi.s    cdone
003a 4230            clr.b    0(a0,d2.w)
003e d441            add      d1,d2
0040 60f2            bra.s    crossout
*          count++;
0042 5243 cdone      addq     #1,d3
0044 5240 loopend    addq     #1,d0           i++ in for loop
0046 60d6            bra.s    mainloop
*      }
0048 51cc done       dbra     d4,iterloop
004c 4281            clr.l    d1              no error
004e=4e40            os9      F$Exit          all finished!

 00000052            ends

Errors: 00000
Memory used: 15k
Elapsed time: 9 second(s)

                     nam      sieve2
                     ttl      primes          by sieve benchmark

* sieve
*
* benchmark, in 68000 assembler
* finds primes using sieve of Eratosthenes
*
* 2nd version - attempt at some optimization
* written to use 'C' start module

                     opt      n
                     opt      f

                     psect    Sieve,0,0,0,0,0

 00001ffe size       equ      8190            flags array size - 1
 00000800 lsize      equ      size/4+1        array size in long words
 00000064 maxiter    equ      100             number of iterations

                     vsect
 00000000 flags      ds.l     lsize
 00000000            ends

* Register usage - Variable names are from 'C' program
*
*   D0.w = i      - main index in flags array
*   D1.w = prime  - current prime number
*   D3.w = count  - count of primes found
*   D4.w = iter   - main iteration loop counter
*   D7.l = scratch for fill loop
*
```

```
*   A0.l = pointer to start of flags array
*   A1.l = pointer to flags[i]
*   A2.l = pointer to flags[k]
*   A3.l = pointer to flags[size]

0000 41ee main:      lea      flags(a6),a0    point to flags array
0004 47e8            lea      size(a0),a3     point to flags[size]
0008 7863            moveq    #maxiter-1,d4
000a 4243 iterloop   clr      d3              count = 0
000c 303c            move     #lsize,d0
0010 2e3c            move.l   #$01010101,d7
0016 2248            move.l   a0,a1
0018 22c7 fillloop   move.l   d7,(a1)+        set all flags[i] true
001a 51c8            dbra     d0,fillloop
001e 70ff            moveq    #-1,d0          start i at -1
0020 2248            move.l   a0,a1           start flags[i] pointer
0022 5240 mainlp1    addq     #1,d0           ++i
0024 4a19            tst.b    (a1)+           search for next prime
0026 6716            beq.s    mainlp2
0028 3200            move     d0,d1           found - prime = i + i + 3
002a d241            add      d1,d1
002c 5641            addq     #3,d1
002e 45f1            lea      -1(a1,d1.w),a2 point to flags[k=i+prime]
0032 6004            bra.s    crosslp2
0034 4212 crosslp1   clr.b    (a2)            clear out flag[k]
0036 d4c1            adda.w   d1,a2           k += prime
0038 b5cb crosslp2   cmpa.l   a3,a2
003a 63f8            bls.s    crosslp1
003c 5243            addq     #1,d3           count++
003e b3cb mainlp2    cmpa.l   a3,a1
0040 63e0            bls.s    mainlp1
0042 51cc            dbra     d4,iterloop
0046 2f03            move.l   d3,-(a7)        printf("%d primes\n",count);
0048 487a            pea      msg(pc)
004c=6100            bsr      printf
0050 508f            addq.l   #8,a7
0052 4e75            rts

0054 2564 msg        dc.b     "%d primes",13,0

 00000060            ends

Errors: 00000
Memory used: 15k
Elapsed time: 7 second(s)
```

READING NON-FLEX BASED DISKS

CONTINUED FROM LAST MONTH

```
                                * Encountered a tab character. If outputting
                                * to a file, the tab must be converted to the
                                * proper spaces, depending on the current
                                * column.
                                *
C468 34    04                   PSHS   B            save current column ctr
C46A C4    07                   ANDB   #%00000111 B = mod(curcol,8)
C46C 86    08                   LDA    #8
C46E 34    04                   PSHS   B
C470 A0    E0                   SUBA   0,S+         A = B - mod(curcol,8)
C472 34    02                   PSHS   A            save this value too
C474 1F    89                   TFR    A,B          B = #spaces to insert

C476 86    20                   LDA    #SP
                C478  PB5       EQU    *
C478 BD    CD18                 JSR    PUTCHR       output a space
C47B 5A                         DECB                until all done
C47C 26    FA                   BNE    PB5

C47E 35    04                   PULS   B            get back # spaces
C480 EB    E0                   ADDB   0,S+         add to cur column ctr
C482 20    BB                   BRA    PB1          and continue

                C484  PB4       EQU    *
C484 BD    CD18                 JSR    PUTCHR       output a char as is
C487 5C                         INCB                incr col ctr
```

```
C488 20    85                  BRA    PB1       continue
                C48A  PB3      EQU    *
C43A 35    36                  PULS   A,B,X,Y   restore regs
C48C 39                        RTS              and return
                      *
                      * Data area for PRTBLK (inited by COPYIT)
                      *
C48D                  EOFSEC   RMB    1
C48E                  EOFOFF   RMB    1
C48F                  EOBUF    RMB    2


                      *
                      * Name     - GETGRN
                      * Function - This routine takes in a trk/sector
                      *            value in D, and reads the granule
                      *            starting at that position. A granule
                      *            has GRANUL sectors with SSIZS bytes
                      *            per sector.
                      *
                      *            On exit, acc D contains the next
                      *            trk/sec after the granule just read.
                      *
                      *            All registers are preserved
                      *
                C491  GETGRN   EQU    *
C491 34    30                  PSHS   X,Y       save registers
C493 32    E9 FFFD             LEAS   -LS6,S    allocate local storage

C497 1F    01                  TFR    D,X       save trk/sec to X

C499 C6    05                  LDB    #GRANUL   # sectors per granule
C49B E7    E9 0000             STB    TMP1,S    save value
C49F 108E  0518                LDY    #BUFFER   where to put data

C4A3 1F    10                  TFR    X,D       transfer trk/sec to D
                C4A5  GB1      EQU    *
C4A5 ED    E9 0001             STD    TMP3,S    save trk/sec
C4A9 8E    C840                LDX    #SYSFCB   get an FCB
XC4AC 17   0027                LBSR   READSS    and read the trk/sec
C4AF 1026  FCD0                LBNE   ERR02     leave if error
                      *
                      * Transfer bytes just read
                      *
C4B3 30    88 40               LEAX   FCB68,X   "from" field
C4B6 CC    0100                LDD    #SSIZS    len to move
XC4B9 17   0024                LBSR   MVC       move it
C4BC 31    AB                  LEAY   D,Y       incr buf addr

C4BE EC    E9 0001             LDD    TMP3,S    get current trk/sec
C4C2 5C                        INCB             goto next sector
C4C3 C1    09                  CMPB   #MAXSEC   hit end of track?
C4C5 23    02                  BLS    GB2       brnch if not
C4C7 4C                        INCA             goto next track
C4C8 5F                        CLRB             sector 0
                C4C9  GB2      EQU    *
C4C9 6A    E9 0000             DEC    TMP1,S    gotten all sectors?
C4CD 26    D6                  BNE    GB1       brnch if not

C4CF 32    E9 0003             LEAS   LS6,S     free local storage
C4D3 35    30                  PULS   X,Y       restore regs
C4D5 39                        RTS              and return
                      *
                      * Data area for GETGRN
                      *
                C4D6  LPC      SET    *
0000                           ORG    $0000
                0000  LCL      SET    *
0000                  TMP1     RMB    1
0001                  TMP3     RMB    2
                0003  LS6      EQU    *-LCL     len of local storage

C4D6                           ORG    LPC       restore PC


                      *
                      * Name     - READSS
                      * Function - This routine reads in the track/sector of
                      *            the disk in the drive specified by an FCB
                      *            pointed to by X. Reg D specifies the
                      *            track/sector to read in.
                      *
                      *            A BNE or BEQ should be used after calling
                      *            this routine to check for possible read
                      *            errors.
                      *
                      *            All registers except A are preserved.
                      *
                C4D6  READSS   EQU    *
C4D6 ED    88 1E               STD    FCBCP,X   set trk/sec
C4D9 86    09                  LDA    #DRSS     set function code
C4DB A7    84                  STA    FCBFC,X   set code in FCB
C4DD 7E    D406                JMP    FMSCAL    read and return


                      *
                      * Name     - MVC
                      * Function - This routine moves the "from" string
                      *            to the "to" string as follows:
                      *
                      *              X -> "from" field
                      *              Y -> "to" field
                      *              D = # characters to move
                      *
                      *            Regs A,B,X,Y are preserved
                      *
                C4E0  MVC      EQU    *
C4E0 34    36                  PSHS   A,B,X,Y   save regs

                C4E2  MOVE     EQU    *
C4E2 34    06                  PSHS   D         save len to move
C4E4 E6    80                  LDB    0,X+      get a "from" char
C4E6 E7    A0                  STB    0,Y+      trans to "to" field
C4E8 35    06                  PULS   D         get len
C4EA 83    0001                SUBD   #1        decr by 1
C4ED 1083  0000                CMPD   #0        done yet?
C4F1 26    EF                  BNE    MOVE      brnch if not

C4F3 35    36                  PULS   A,B,X,Y   restore regs
C4F5 39                        RTS              and return


                      *
                      * Name     - CLC
                      * Function - This routine compares two strings
                      *            as follows:
                      *
                      *
                      *
                      *              X -> string1
                      *              Y -> string2
                      *              A = length of strings to compare
                      *
                      *            String1 is compared to String2, and the
                      *            appropriate condition code is set.
                      *
                      *            Regs A,B,X,Y are preseved.
                      *
                C4F6  CLC      EQU    *
C4F6 34    36                  PSHS   A,B,X,Y   save regs

                C4F8  COMPR    EQU    *
C4F8 E6    80                  LDB    0,X+      get a string1 char
C4FA E1    A0                  CMPB   0,Y+      compare to string2 char
C4FC 26    03                  BNE    RPROC     if neq then done

C4FE 4A                        DECA             all through strings?
C4FF 26    F7                  BNE    COMPR     brnch if not

                C501  RPROC    EQU    *
C501 35    36                  PULS   A,B,X,Y   restore regs
C503 39                        RTS              and return


                      *
                      * Name     - PDATA
                      * Function - This routine prints a string to an
                      *            output device. On entry, X points
                      *            to the string to print. As with the
                      *            PSTRNG routine in FLEX, an EOT
                      *            delimiter must appear after the
                      *            string to print.
                      *            Regs A and B will remain as is, and
```

```
                   *           X will be pointing to the EOT
                   *           delimiter char.
                   *
          C504  PDATA   EQU    *
C504 34   06            PSHS   A,B       save regs

          C506  PDATA1  EQU    *
C506 A6   80            LDA    0,X+      get char from string
C508 81   04            CMPA   #EOT      end of text?
C50A 27   05            BEQ    PDATA2    brnch if yes
C50C BD   CD18          JSR    PUTCHR    print char
C50F 20   F5            BRA    PDATA1    and continue

          C511  PDATA2  EQU    *
C511 35   06            PULS   A,B       restore regs
C513 39                 RTS              and return


                   *
                   * Name     - ROUTE
                   * Function - This routine determines where the user
                   *            wants his output to be routed. The user
                   *            has three options:
                   *
                   *            RO=TERMINAL: In this case, no control
                   *              info need be set, other than ensuring
                   *              the output switch (OSWTCH) is non
                   *              zero so FLEX will use the OUTCH2
                   *              routine to print characters.
                   *
                   *           RO=PRINTER: In this case, the printer
                   *              module routine must be loaded into
                   *              memory if it is not already there,
                   *              and the output switch must be cleared
                   *              so FLEX will use OUTCH to process
                   *              the characters
                   *
                   *            RO=FILE: In this case, the user is
                   *              prompted for a file to which all
                   *              output will be routed. FLEX's PUTCHR
                   *              routine automatically routes outputted
                   *              chars to the file's FCB if told
                   *              to do so.
                   *
                   *          All registers are preserved
                   *
          C514  ROUTE   EQU    *
C514 34   36            PSHS   A,B,X,Y   save regs

C516 8E   0000          LDX    #0        clear file output address
C519 BF   CC24          STX    FOA       in FLEX

C51C 8E   0223          LDX    #OUTFIL   prompt user for route
C51F BD   CD1E          JSR    PSTRNG
C522 BD   CD09          JSR    INCH      get response
C525 84   5F            ANDA   #$5F      lowercase->uppercase

C527 81   50            CMPA   #'P       printer?
C529 27   6F            BEQ    R2        brach if so

C52B 81   54            CMPA   #'T       terminal?
C52D 27   72            BEQ    R3        brnch if so

C52F 81   0D            CMPA   #CR       CR? (same as terminal)
C531 27   6E            BEQ    R3        if so brnch

C533 81   46            CMPA   #'F       file?
C535 26   71            BNE    R6        leave if not

                   *
                   * output to a file
                   *
C537 8E   0257          LDX    #TOFILE   ask for "to" filename
C53A BD   CD1E          JSR    PSTRNG
C53D BD   CD1B          JSR    INBUF     get response
C540 8E   0306          LDX    #FILFCB   "to" FCB
C543 BD   CD2D          JSR    GETFIL    validate filename
C546 24   08            BCC    R4        check if file exists

C548 8E   0281          LDX    #INVSPC   invalid filename entered
```

```
C54B BD   CD1E          JSR    PSTRNG
C54E 20   58            BRA    R6        return

          C550  R4      EQU    *
C550 86   08            LDA    #OUT      set extension up
C552 BD   CD33          JSR    SETEXT
                   *
                   * Open the file
                   *
C555 86   02            LDA    #XOWRIT   open for write
C557 A7   84            STA    FCBFC,X   save in FCB
C559 BD   D406          JSR    FMSCAL    call FMS
C55C 27   08            BEQ    R7        continue if file not there
                   *
                   * File already exists -> say so and return
                   *
C55E 8E   0290          LDX    #FXIST
C561 BD   CD1E          JSR    PSTRNG
C564 20   42            BRA    R6        return

          C566  R7      EQU    *
C566 8E   0288          LDX    #FTYPE    ask for file type
C569 BD   CD1E          JSR    PSTRNG
C56C BD   CD09          JSR    INCH      get response
C56F 84   5F            ANDA   #$5F      lowercase->uppercase

C571 5F                 CLRB
C572 81   54            CMPA   #'T       text?
C574 27   11            BEQ    R8        brnch if so

C576 81   0D            CMPA   #CR       CR? (same as text)
C578 27   0D            BEQ    R8        brnch if so

C57A 81   42            CMPA   #'B       binary?
C57C 26   2A            BNE    R6        bad leave if not

C57E C6   A0            LDB    #OPN+BINRY set open-bin bits
C580 86   FF            LDA    #SCFMSC   type=binary
C582 B7   0411          STA    FILFCB+FCBSCF set compress. flag
C585 20   02            BRA    R1

          C587  R8      EQU    *
C587 C6   C0            LDB    #OPN+TEX  set open-text bits
          C589  R1      EQU    *
C589 8E   0306          LDX    #FILFCB   set all o/p to file
C58C BF   CC24          STX    FOA
C58F 7F   CC22          CLR    OSWTCH    o/p to the file
C592 F7   0519          STB    OPNFLG    save flag bits
C595 7F   CC04          CLR    WIDTH     zero TTY width value
C598 20   0A            BRA    R9        return

          C59A  R2      EQU    *
                   *
                   * perform setup for output to printer
                   *
C59A 7F   CC22          CLR    OSWTCH    o/p to aux. device
C59D 8D   0E            BSR    PRTSET    fetch printer module

C59F 20   03            BRA    R9        and return

          C5A1  R3      EQU    *
                   *
                   * perform setup for output to terminal
                   *
C5A1 7C   CC22          INC    OSWTCH

          C5A4  R9      EQU    *
C5A4 1C   FE            CLC              set good RC
C5A6 20   02            BRA    R10       and leave

          C5A8  R6      EQU    *
C5A8 1A   01            SEC              set bad RC
          C5AA  R10     EQU    *
C5AA 35   36            PULS   A,B,X,Y   restore regs
C5AC 39                 RTS              and return


                   *
                   * Name     - PRTSET
                   * Function - This routine is called to load the
                   *            printer module if necessary.
```

```
                  *          and init the PIA, so that output can
                  *          be routed to the printer.
                  *
                  *       No registers are preserved
                  *
            C5AD  PRTSET  EQU    *
C5AD 7F    CC09           CLR    PAU        disable pause feature
C5B0 B6    CCE4           LDA    POUT       get 1st byte of space
C5B3 81    39             CMPA   #$39       is it "RTS"?
C5B5 26    29             BNE    P15        if not the loaded

                  *
                  * Load printer routine
                  *
C5B7 8E    02F4           LDX    #PSYS      move in print rtn name
C5BA 108E  C844           LDY    #SYSFCB+FCBNAM into system FCB
C5BE CC    000B           LDD    #11
C5C1 17    FF1C           LBSR   MVC
C5C4 7F    C843           CLR    SYSFCB+FCBDN check drive 0

C5C7 8E    C840           LDX    #SYSFCB    point to system FCB
C5CA 86    01             LDA    #XOREAD    open for read
C5CC A7    84             STA    FCBFC,X
C5CE BD    D406           JSR    FMSCAL     call FMS
C5D1 27    05             BEQ    P1         brnch if open ok

C5D3 BD    CD3F           JSR    RPTERR     report error
C5D6 20    11             BRA    P2         and return

            C5D8  P1      EQU    *

C5D8 86    FF             LDA    #SCFNSC    set for binary read
C5DA A7    88 3B          STA    FCBSCF,X
C5DD BD    CD30           JSR    LOAD       load module

            C5E0  P15     EQU    *
C5E0 BD    CCC0           JSR    PRINIT     go init port
C5E3 8E    CCE4           LDX    #POUT      get o/p address
C5E6 BF    CD10           STX    OUTCH+1    stuff in FLEX
            C5E9  P2      EQU    *
C5E9 39                   RTS               return


                  *
                  * Name     - RESET
                  * Function - This routine is called to reset the
                  *            FLEX output switch and close any file
                  *            that might be open through the FILFCB
                  *            FCB.
                  *
            C5EA  RESET   EQU    *
C5EA 86    01             LDA    #1
C5EC B7    CC22           STA    OSWTCH     reset output switch
C5EF B7    CC09           STA    PAU        enable pause
C5F2 7D    0519           TST    OPNFLG     is a file open?
C5F5 27    15             BEQ    RSX        return if not
                  *
                  * Output was going to file, close it
                  *
C5F7 B6    051A           LDA    TTYWID     restore TTY width
C5FA B7    CC04           STA    WIDTH

C5FD 8E    0306           LDX    #FILFCB    set FCB
C600 86    04             LDA    #XCLOSE    close code
C602 A7    84             STA    FCBFC,X
C604 BD    D406           JSR    FMSCAL     call FMS
C607 27    03             BEQ    RSX        if ok return

C609 BD    CD3F           JSR    RPTERR     report error
            C60C  RSX     EQU    *
C60C 7F    0519           CLR    OPNFLG     indicate no file open
C60F 39                   RTS               and return


                          END    START
SYMBOL TABLE:

ADDBX  CD36  ASREAD 0001  ASWRIT 0002  BAC    0008  BADFIL 026C
BADIN  03B6  BAK    0005  BAS    0003  BELL   0007  BIN    0000
BINRY  0020  BS     CD00  BSE    CC07  BSIZS  0500  BUFFER 051B
BUFPNT CC14  CLASS  CD21  CLC    C4F6  CLN    CC1A  CLOCK  F700
```

```
CMD    0002  CMDFLG CC28  COC    CC29  COLDS  CD00  COMPR  C4F8
COPYIT C2C8  CR     000D  CRLF   000A  CTR1   0002  CURC   CC18
DAT    0007  DDRV   DE00  DEL    CC01  DEPTH  CC03  DIR    0009
DIRBEG C516  DIRMSG 014C  DIRPTR 0000  DIRSIZ 0002  DIRTS  0005
DOCMND CD4B  DOS    CD00  DP1    C305  DP2    C33D  DP3    C320
DP4    C321  DRV    0518  DRVPNT 0102  EJECT  CC08  ENDCOP 01A4
ENDDIR 016B  ENTADR 0000  ENV    CC2D  EOBUF  C48F  EOFUFF C48E
EOFSEC C480  EOL    CC02  EOT    0004  ERR    C179  ERRO2  C173
ESC    CC0A  ESCRR  CC16  FACP   0010  FADP   0040  FARP   0020
FAMP   0080  FCBAS  0002  FCBASE D409  FCBCDA 002F  FCBCP  001E
FCBCRN 0020  FCBCUR D40B  FCBDI  0022  FCBDN  0003  FCBEDA 0013
FCBESB 0001  FCBFA  000F  FCBFC  0000  FCBFCD 0019  FCBFUD 0032
FCBFS  0015  FCBFSN 0017  FCBLEN 0140  FCBLP  001C  FCBNAM 0004
FCBNUB 0024  FCBRI  0023  FCBRSI 0010  FCBRSZ 0018  FCBSB  0040
FCBSCF 003B  FCBSDR 0035  FCBSDA 0011  FCBVER D435  FCDDAY 001A
FCDMTH 0019  FCDYR  0018  FDE    0000  FDEAPW 0012  FDEEN  0000
FDEFIF 0000  FDEFN  0005  FDEIGN 0001  FDELEN 0020  FDELRL 0004
FDEND  001E  FDEOFM 0015  FDEUFL 0003  FDEOFN 0014  FDERSV 0002
FDEUPW 0010  FDEXEL 0016  FIA    CC26  FIEF   CC2F  FIFINV 0008
FIFPOE 0080  FIFREE 0010  FIFSYS 0040  FILFCB 0306  FINDIR C22D
FLEX   CD00  FMS    D400  FMSCAL D406  FMSCLS D403  FMSERR CC20
FMSINT D400  FOA    CC24  FOUND  C403  FSIZ   0000  FSMRAM 0002
FSMSED 0000  FTYPE  02BB  FXIST  0290  GB1    C4A5  GB2    C4C9
GET1   C353  GET2   C365  GET3   C367  GET4   C361  GETCHR CD15
GETDRV C345  GETFIL CD2D  GETBRN C491  GETHEX CD42  GE10   C3F4
GETSIZ C26E  GETXEP C2A3  GRANUL 0005  GS     C28C  GS2    C29B
GT1    C38A  GT2    C39B  GT3    C3A1  GTBAD  C3A6  GTGOOD C3A7
GTSPEC C36A  GTRETN C2C5  GXIIT  C2C4  HEAD   0100  INBUF  CD1B
INCH   CD09  INCH2  CD0C  INDEC  CD48  INDRV  0208  INFILE 01F4
INTRO  02FF  INVSPC 0281  IOFLG  CC21  ISWTCH CC23  JT     C13B
LAD    CC1B  LCL    0000  LF     000A  LNEBUF C080  LOAD   CD30
LPC    C404  LS1    0007  LS2    0003  LS3    0014  LS5    0005
LS6    0003  LSTRM  CC11  M0     C143  M1     C148  M2     C155
M3     C160  MAP    CD00  MAXSEC 0009  MEMEND CC2B  MEM2   C11C
MENU   C119  MOVE   C4B2  MVC    C4E0  MX     C163  NULL   CC05
NXTCH  CD27  OPN    0080  OPNFLG 0519  OSWTCH CC22  OUT    0008
OUTADR CD45  OUTCH  CD0F  OUTCH2 CD12  OUTDEC CD39  OUTDRV 0167
OUTFIL 0223  OUTHEX CD3C  OUTSIZ 0004  P1     C5D8  P15    C5E0
P2     C5E9  PAU    CC09  PB1    C43F  PB2    C41C  PB3    C48A
PB4    C484  PB5    C47B  PB6    C430  PB7    C425  PB8    C43B
PB9    C46A  PCRLF  CD24  PD1    C102  PD3    C18C  PD4    C20F
PD5    C1E0  PD6    C1F7  PDATA  C504  PDATA1 C506  PDATA2 C511
PDIR   C17F  PDXIT  C225  POUT   CCE4  PRCHK  CCD8  PREVC  CC19
PRINIT CCC0  PROM   C125  PRT    000A  PRTBLK C410  PRTSET C5AD
PRTTEL C247  PSTRNG CD1E  PSYS   02F4  PUTCHR CD18  PX1    C254
R1     C589  R10    C5AA  R2     C59A  R3     C5A1  R4     C550

R6     C5A8  R7     C566  R8     C5B7  R9     C5A4  RDERR  02E0
READSS C4D6  RENTER CD06  RESET  C5EA  RET    C409  ROUTE  C514
RPNOC  C501  RPTERR CD3F  RSTRID CD2A  RSX    C60C  SBDATA 0044
SBLINK 0040  SBRSI  0042  SCFNSC 00FF  SCFSC  0000  SCR    0006
SETEXT CD33  SETPAU C168  SFA    C980  SIRCRE 0023  SIRDAY 0024
SIRFSB 001D  SIRFSE 001F  SIRFSS 0021  SIRLEN 0028  SIRMTH 0023
SIRNTS 0026  SIRNAM 0010  SIRTS  0003  SIRVOL 0018  SIRYR  0025
SP     0020  SP4    01EF  SPS    C700  SRC1   C38C  SRC2   C303
SRCA   C3E7  SRCHR  C3AA  SRCXIT C3FF  SSIZS  0100  START  C100
START1 C103  STAT   CD4E  STKA   C000  SVEFIL 0000  SVDR   CC0E
SYDRV  CC0B  SYS    0004  SYSCON CC4E  SYSCR1 CC00  SYSCR2 CC2A
SYSCR3 CC30  SYSCR4 CCFB  SYSFCB C840  TAB    CC06  TABCON 0009
TDSIZ  0002  TEMP   0002  TEX    0040  TFSIZ  0008  TMP1   0000
TMP3   0001  TMP4   0000  TMP5   0006  TMPXEL 000C  TNOTS  0003
TOFILE 0257  TRADDR CC1E  TRFLG  CC1D  TTYWID 051A  TXT    0001
UCA    C100  UCTA   CC12  URAM   0000  WARMS  CD03  WELCOM 012B
WIDTH  CC04  WKDRV  CC0C  XBOR   0016  XCLOSE 0004  XDELET 000C
XDLDEC 00FE  XELEND 00FF  XFND   0014  XGIR   0007  XGRB   0011
XMSS   000F  XOBIR  000A  XOREAD 0001  XOSIR  0010  XOUPUT 0003
XOWRIT 0002  XPIR   0008  XPOSN  0015  XPRB   0012  XRENAM 0000
XRES1  000B  XRES2  000E  XRES3  0013  XREWND 0005  XRSS   0009
XRWNB  0000  XWBS   000A
```

COMMUNICATING WITH OS/9

by Sidney Thompson
181 Greenbriar Court
Conyers, GA 30208
404-922-3097

and Bud Pass
1454 Latta Lane
Conyers, GA 30207
404-483-1717

INTRODUCTION

The following describes the application of OS/9 to the task of communicating (usually over modems and telephone lines) with other computer systems and with computer terminals.

Because telephone lines are inherently noisy, one of the major requirements which must be placed on communications hardware and software is error control and correction.

Because many computer systems and many terminals have unusual and different requirements, the communications software must be as flexible and as transparent as possible.

The program described below (named CMODEM) attempts to provide these facilities under OS/9, to allow communication with other systems, in a potentially error-free environment.

The program was written to overcome the many problems involved with trying to transfer data between not only various machines but between different operating systems on the same machine.

CMODEM FEATURES

CMODEM contains a combination of features found in the original BDS-C Teledit program, a TTY program published in System 68, Yam (Yet Another Modem), and other features added for this version only ("I love it, but could you make it do ...? ").

It uses the Ward Christiansen MODEM file transfer protocol to facilitate the transfer of files between systems that support this protocol. Both MODEM and CMODEM support a checksum block verification that allows for error-free transfer of data between two computers supporting the same protocol.

For those systems that do not support a batch transfer protocol, CMODEM supports an ASCII file collection/dump mode for transfer of source files.

The use of multiple file transmission modes allows sending and receiving either binary or text files.

CMODEM will work at speeds of at least 1200 Baud for transfer of files at high rates of speed, for those systems with ACIA's or other asychronous communications support devices. It is used regularly at 4800 Baud between adjacent machines to transfer data between them. The COCO is capable of reliable transmission at rates of up to 2400 Baud, in general, and may (in some cases) be capable of transmission at rates of up to 4800 Baud or higher, when using the PIA printer port.

CMODEM is written in a high level language (C) and is available in both source and object formats to allow adaptation to other systems or for the addition of new options for specific requirements. It uses the same source code for both the FLEX and OS/9 versions. This facilitates the transfer of files between FLEX and OS/9 systems, in addition to the ability of communicating with CP/M systems, as described above.

The FLEX version of CMODEM allows the user to change the modem port address at execution time, since there is no standard address for a modem, as there is (normally) for the console. The OS/9 version of CMODEM uses a standard OS/9 path descriptor name (by default, "/T1") to access the modem port; this is compatible with both OS/9 level I and OS/9 level 2. However, due to the overhead associated with COCO OS/9, a special version of CMODEM containing its own drivers is used for a COCO with a PIA printer/modem port.

Since it is inconvenient to modify system parameters upon each entry to the program, CMODEM supports the use of a parameter file (profile) which can be used to define the modem port address or name, baud rate of the modem port (for those operating systems and devices supporting variable modem port baud rate), the duplex status required (half or full), etc. This parameter file is described later in this discussion. It also allows the definition of an expert mode, which allows the use of CMODEM without the occasional display of the menu. This feature is useful when one has used CMODEM long enough that the menu has become completely memorized and it decreases the time required to transfer among the different operating modes.

The current FLEX version of CMODEM does direct I/O to the modem port and will not work with those systems not using ACIA's; however, a special version is available for COCO FLEX, just as a special version is available for COCO OS/9.

CMODEM OPERATIONS

CMODEM is menu-driven to make it as simple as possible to use. The menu is displayed upon entry into the program and also when the user strikes the defined ATTENTION key.

The selectable program modes are described below.

T: Terminal mode - no text collection

CMODEM simulates an ASCII terminal in this mode.

By default, eight-bit characters are sent and received without parity. The DEL character (7F hex) is the only character that will be removed from the conversational terminal mode input data stream. Most other characters are passed unchanged to the console, so formatting characters and nonsense noise characters are usually passed thru for the user's inspection.

To return to the selection menu, the user may hit the SPECIAL character. The current SPECIAL character of <ctrl>-@ (NULL) was chosen because it is unlikely to be struck accidentally, or to be required input to most systems. Changing the SPECIAL character requires recompiling CMODEM with the desired character code in the following statement:

```
#define SPECIAL ...
```

A BREAK key is also supported for those systems that require an actual break signal be placed on the line. This is set by default to <ctrl>-underline, but may be changed to any desired key. It must be emphasized that this key currently works only on a SWTPC-compatible system using a 6850 ACIA under FLEX and for a COCO system under FLEX or OS/9.

?: Toggle expert mode

This command flips expert mode, alternately outputting and not outputting the menu to the console. This saves time for those very familiar with using the program.

G: Gather text in memory buffer

CMODEM processes text received in this mode similar to that received in terminal mode, except that any text characters received from the communications link are saved into a text buffer. Tab, carriage-return, line-feed, and form-feed characters are placed into the buffer; any other control characters are discarded. The one exception to this is when an end of line CR LF sequence is received, the program only inserts the CR into the memory buffer. This prevents the storage of double-spaced text to the disk file which could occur if both control characters are placed into the data file. If the text buffer becomes full, CMODEM will issue an X-OFF to the remote system. The program then will continue to capture data for a maximum of an additional 95 characters. This allows time for the remote system to receive the X-OFF and to stop sending data.

If the remote system at the other end of the line does not honor the X-OFF, CMODEM will flush the buffer to the disk storage file and then return to accept more data from the communications line. Data arriving during this buffer-dumping process will almost certainly be lost. In this case, since this loss is not normally desirable, the amount of data transmitted in each group should be limited to less than the number of characters required to fill the text buffer.

If, on the other hand, the remote system does honor X-ON/X-OFF, no data will be lost while CMODEM flushes the memory buffer to disk. Upon resuming its data capture mode, CMODEM issues an X-ON to restart the remote transmission.

There are some systems which either do not recognize X-ON/X-OFF or require characters such as the ESCape to do both the halt and resume function. Using this option of CMODEM on these systems will require either that the values within the program for STOP and RESUME be modified.

Since the standard printer port on a COCO works in half-duplex mode only, transmitting the X-OFF will garbage any characters being received. The user must be aware of this fact and request the retransmission of the garbaged data or somehow arrange for the gather buffer never to be overflowed.

V: View text buffer

This mode allows the viewing of the data which currently is in the memory capture buffer. This is helpful when one wishes to review a long transmission from a remote system; although this mode allows only paging forward, the view process may be restarted at the beginning, as required. On non-COCO systems, the carriage return displays the next line, the space bar displays the next page, and the ATTENTION key returns the menu. On COCO systems, the space bar toggles a pause and the ATTENTION key returns the menu.

The memory capture buffer and the batch transfer buffer share the same internal memory area. This means that if one initiates a batch file transfer, any data which may be in the buffer will be destroyed during the process.

A: Enter ASCII file dump mode

CMODEM will transfer a text file over the communications line. It can be used to send a file to those systems that do not support the MODEM protocol, but do support paper tape capture. This mode honors the X-ON/X-OFF protocol, so that the remote system may control the flow of data during the process.

K: Kill text buffer

This command marks the text buffer as empty and exits ASCII text capture mode. It is normally used when the user has initiated capture mode, but has captured unexpected data and does not wish to save it on disk.

E: Toggle Echo (duplex) mode

This option determines whether characters input from the console are echoed back to the display screen locally or not.

It should be toggled to "full" if the user is communicating in echoplex mode and is receiving an echo from the remote station.

It should be toggled to "half" if the user is working with a system which does not echo to the sender or is in half duplex mode.

The current setting of this option is displayed in the menu.

M: Modify program option parameters

The non-COCO, FLEX version of CMODEM is able to modify several of the current program parameters with this option. A submenu of current options guides the entry of program parameter changes. These include the following parameters:

port address for modem
baud rate of the modem port
parity settings of the modem port

B: Set Baud rate

The COCO versions of CMODEM are able to set the Baud rate of the attached PIA printer/modem port with this option. A submenu of available Baud rates guides the entry of program parameter changes. Although Baud rates higher than 2400 Baud appear in the menu, they are not reliable enough, in general, for normal usage.

D: Set Delay rate

The COCO versions of CMODEM are able to "tweak" the Baud rate of the attached PIA printer/modem port with this option. A submenu of available Baud rates guides the entry of program parameter changes. Although Baud rates higher than 2400 Baud appear in the menu, they are not reliable enough, in general, for normal usage. Since the PIA printer/modem port is driven in a bit-banging mode by software, it is not as tolerant of Baud rate variations as a hardware device such as an ACIA. This option allows the variation of the standard delay values to attempt to reduce communications errors.

S: Send a file with MODEM protocol

CMODEM prompts for the name of the file to send, then waits for the receiving computer (which must be using this program or one that uses the same protocol) to get into synchronization. The details of the protocol are provided later.

CMODEM returns to Terminal mode after completion of this option, whether the transfer was successful or not.

R: Receive a file with MODEM protocol

CMODEM prompts for the name of the file to receive, then waits for the sending computer (which must be using this program or one that uses the same protocol) to get into synchronization. The details of the protocol are provided later.

CMODEM returns to Terminal mode after completion of this option, whether the transfer was successful or not.

X: Binary file transfer with MODEM protocol

CMODEM prompts for the name of the file to send, then waits for the receiving computer (which must be using this program or one that uses the same protocol) to get into synchronization. The details of the protocol are provided later.

CMODEM returns to Terminal mode after completion of this option, whether the transfer was successful or not.

This option may be used to transfer compiled binary files from one system to another like system. It can also be used between FLEX systems to send files in space-compressed mode to save transmission time.

This file transfer does not involve character transliteration of the contents of the file being transmitted. There is no conversion of CR to NL, CR to CRNL, or CRNL to CR. This may cause unexpected results, depending upon the system from which a file is being received or to which it is being sent.

Y: Binary file receive with MODEM protocol

CMODEM prompts for the name of the file to receive, then waits for the sending computer (which must be using this program or one that uses the same protocol) to get into synchronization. The details of the protocol are provided later.

CMODEM returns to Terminal mode after completion of this option, whether the transfer was successful or not.

This option may be used to transfer compiled binary files from one system to another (like) system. It can also be used between FLEX systems to send files in space-compressed mode to save transmission time. This file transfer does not involve character transliteration of the contents of the file being received.

When this option is used under OS/9, the file must then be processed with ATTR to make the program executable at the receiving end, assuming an executable OS/9 module was transmitted successfully. This is true because the file attributes are associated with the disk directory, and not with the contents of the file.

Q: Quit

For non-COCO, FLEX versions of CMODEM, this option resets the modem port, dropping the telephone line or other equipment using the RTS signal on an ACIA as a DTR indication. For all versions, it returns to the operating system.

O: Operating system return

For non-COCO FLEX versions of CMODEM, this option returns to the operating system without resetting the modem port. This can be used to return to the operating system to perform various functions without disturbing the equipment using the RTS signal on the ACIA as a DTR indication.

H: Hang up the phone line

This option is used to hang up the phone line by dropping the DTR(RTS) signal on the modem port, on FLEX versions of CMODEM using ACIA's. This is very useful for the auto-dial type modems in which the user may wish to hang up from one system but is not ready to exit from the program. The port is reset after approximately one second so that it is then available to call another remote system.

F: Flush data collection buffer to capture file

CMODEM flushes the text buffer to disk and resets the memory pointer to the beginning of the buffer. This would be used if it were desirable to capture several small files and concatenate them onto disk or to capture files longer than the available buffer size. The capture file is currently closed on exit from the program or upon receipt of a Close command, so that the buffer may be flushed multiple times to the same file.

C: Close text collection file

CMODEM flushes the memory buffer to the collection disk file and then closes the file. This allows the capture of multiple files without the necessity of exiting CMODEM.

L: Line feed generated locally for CR

While this may not be the most popular of all options, it proves to be very useful for those systems which send only a carriage return or do not echo anything after an end of line character is sent. This includes the COCO printer port and many other systems. This option allows the production of a local CR NL sequence to prevent repeatedly overwriting the same line with successive lines.

CMODEM PARAMETER FILE (PROFILE)

As described earlier, CMODEM supports a parameter file for the purpose of automatically modifying certain default parameters. For FLEX, this file is named PROFILE.TXT and must be placed on the working drive. For OS/9, this file is named PROFILE and must be placed in the current directory. The user is responsible for creating the parameter file according to the description below. If the file is not available or a parameter is not modified, the default parameter values will be used.

Each record in the parameter file contains a keyword and an optional operand, and is terminated with a carriage return.

For non-COCO FLEX CMODEM, the parameter file has the following contents:

BAUD	300/1200	(1200)	baud rate
DUPLEX	F/H	(F)	duplex switch
MODEM	xxxx	(E074)	modem port address
UNITS	nn	(10)	disk buffers per I/O (1-25)
XPERT		(no)	expert mode toggle

For non-COCO OS/9 CMODEM, the parameter file has the following contents:

DUPLEX	F/H	(F)	duplex switch
PORT	/aaaa	(/T1)	modem path descriptor name
UNITS	nn	(10)	disk buffers per I/O (1-25)
XPERT		(no)	expert mode toggle

For COCO versions of CMODEM, the parameter file has the following contents:

BAUD	300/1200	(1200)	baud rate
CC110	xxxx	(04CC)	110 baud delay factor
CC300	xxxx	(018F)	300 baud delay factor
CC600	xxxx	(00DE)	600 baud delay factor
CC1200	xxxx	(006C)	1200 baud delay factor
CC2400	xxxx	(0033)	2400 baud delay factor
CC4800	xxxx	(0019)	4800 baud delay factor
CC9600	xxxx	(000E)	9600 baud delay factor
CC19200	xxxx	(0007)	19200 baud delay factor
DUPLEX	F/H	(F)	duplex switch
XPERT		(no)	expert mode toggle

COCO PIA MODEM PORT WIRING

The COCO version of CMODEM requires that the PIA modem port be wired as follows:

pins 1 and 2	received data
pin 3	ground
pin 4	transmitted data

Note that this is different from that used by Tandy and Microware.

CMODEM INSTALLATION

This section is of interest only to those who have purchased the source version of CMODEM.

The source version of CMODEM is composed of nine modules. This is because of the composite size of the various modules and the ability of some editors to work well only with files that can fit into main memory.

The CMODEM modules are as follows:

cmodem.c	option selections
cmodem.inc	common information
cmodterm.c	terminal emulation
cmodsend.c	modem file transfer
cmodrecv.c	modem file receive
cmodfile.c	file open, close. read, write
cmodxmit.c	ASCII file transmission
cmodsupt.c	utility functions
make.c	includes all header files and .asm modules
makefile	compilation command file

The "makefile" file provides a command file which will perform the entire compilation and assembly process on the CMODEM modules. The "makefile" modules are somewhat different among the various versions of the C compilers available under OS/9 and FLEX.

Since CMODEM was rewritten into Dyna-C, the Make.c file is used to include all of the required library files and the assembly language files that are produced by the compiler (.asm). There are also some rewritten library files provided with CMODEM since the standard versions available with Dyna-C did not support all of the features required and did not comply with the standards normally found on UNIX-type machines.

While this may seem to be troublesome, it does make the source to CMODEM usable to more people since it does not require the purchase of one of the $400 C compilers currently available for the 6809. The new version of Microware C for the COCO was not available when this program was being developed.

While conversions have not been attempted, there is no reason that some of the other small C's could not also be used if the proper library functions were written for them. There are versions of CMODEM which have been developed for Microware C and Introl C in the past.

SYSTEM-DEPENDENT DEFINITIONS

This section is of interest only to those who have purchased the source version of CMODEM. It describes some of the "#define"s that are used in the source version of CMODEM. Some of these may require modification to suit the requirements of particular systems.

TWIDTH

This is the width of the terminal display in characters.

TLENGTH

This is the number of horizontal lines that can be displayed on the terminal.

MYMODEM

This is the default address of the ACIA port being used for the modem.

MYTERMINAL

This is the default address used for the console terminal.

LPSEC

This is an internal timing loop value used for time delay.

CPUMHZ

This is the default clock speed for the processor and is used as part of the timing loop formula.

SECSIZ

This is the size of each received or transmitted data sector. It is set to 128 bytes to be compatible with CP/M MODEM transfer protocols.

CAPSIZ

This is the total buffer capacity in bytes. It is the size of each sector times the number of sectors that will be sent or captured before a disk read or write.

SPECIAL

This is the control sequence that is recognized by the program from the console as a request to display the menu. It may be reset but has the default value <ctrl>-@.

BREAK

This is the control sequence that is recognized by the program from the console as a request to transmit a physical break condition. It may be reset but has the default value <ctrl>-underline.

BRKCODE

This is the value that will be sent to the modem ACIA port to initiate a break code on the line.

SYSDEPEND

Since the various C compilers available seem to return different codes for OPEN, this variable is defined as required for a given C compiler. It is defaulted to NULL for Dyna-C.

MODEM PROTOCOL DEFINITION

The following description of the Ward Christiansen MODEM protocol was derived from various sources, including the original MODEM description, the TELENET description, the TTY description, and a description on the NEWSNET by Keith Peterson, with additions for CMODEM.

This description provides the criteria for operation of the MODEM protocol if it is to be compatible with the various CP/M and UNIX systems in operation.

Using this protocol, CMODEM has been used to transfer files from OS/9-based, UNIFLEX-based, and FLEX-based systems to and from each other, to and from CP/M-based systems using MODEM, and to and from UNIX-based systems using TMODEM, UMODEM, and XMODEM. It has also been used to send and receive files from other assembly language MODEM programs, as well as programs written in C, that exist on the various C-Nodes across the country.

The control characters that define the MODEM data block are as follows:

<soh>	01H
<eot>	04H
<ack>	06H
<nak>	15H
<can>	18H

TRANSMISSION-MEDIUM LEVEL PROTOCOL

Following are the attributes of the transmitted and received data:

asynchronous

8 data bits

no parity

one stop bit

This protocol does not place any restrictions on the contents of the data being transmitted. Any type of data can accurately be sent using this protocol. This includes the following types of files:

ASCII text files

binary compiled program files

command files containing control characters

any other arbitrary files

While this protocol is designed for use with an 8-bit data format, it could easily be adapted for use in a 7-bit environment and used for the transmission of ASCII-only (or unpacked-hex) data. This could be done simply by having both the sender and receiver logically mask any protocol-dependent data to clear the eighth bit before transmitting and after receiving it. This would be limited to the variable-data portion of the transferred data, and would not include the following 8-bit protocol control data, described later:

the checksum

the block numbers

the one's-complement of the block number

To maintain compatibility with the CP/M file structure, the data file must follow the following data format:

ASCII tabs must be used (09H); tabs are assumed set every 8 positions;

All lines must be terminated by CR/LF (0DH 0AH);

End-of-file must be indicated by one or more <ctrl>-Z (1AH) characters;

Data is variable length, but while the file is being handled by the transfer program, it should be considered a continuous stream of data bytes. This continuous stream is then broken into 128-byte chunks by the program for transmission and is regrouped upon reception.

If the data ends exactly on a 128-byte boundary, i.e. CR in 127, and LF in 128, an additional block containing the <ctrl>-Z EOF character(s) is optional, but is preferred, as some CP/M utilities and user programs do not handle EOF without <ctrl>-Z correctly.

The last block sent is identical to all the others; there is no "short block" supported by the MODEM protocol. Padding is added to the block to complete the required 128 byte block size on the last data block. While one would be tempted to pad this block with <ctrl>-Z, this will work reliably only with CP/M-based systems. Other systems, such as OS/9, UNIX, FLEX, and, UNIFLEX, would not usually process the <ctrl>-Z characters correctly. It is safer to use spaces for padding even though this could possibly add a slight length to the file, and potentially cause difficulties with binary transmissions to non-CP/M systems.

MESSAGE BLOCK-LEVEL PROTOCOL

The format of every data block transferred in either direction under the MODEM protocol is as follows:

<SOH> <blk #> <255-blk #> <--128 data bytes--> <cksum>

In which:

<soh> = 01 hex;

<blk #> = binary block number (starts at 01H, increments by 01H, and wraps around from 0FFH to 00H;

<255-blk #> = one's-complement of binary block number;

<cksum> = the last eight bits of the sum of the data bytes.

PROGRAM-LEVEL PROTOCOL

COMMON TO BOTH SENDER AND RECEIVER:

All errors are retried 10 times.

Some versions of the protocol (but not CMODEM) use <can> (18H), to cancel the transmission operation. This has never been adopted as a standard, since a single "abort" character makes the transmission susceptible to false termination caused by an <ack> <nak> or <soh> being corrupted into a <can> and canceling transmission.

The protocol may be considered "receiver driven", that is, the sender need not automatically re-transmit, although it does in most of the current implementations, including CMODEM.

RECEIVING PROGRAM CONSIDERATIONS:

The receiver has a 10-second timeout.

It sends a <nak> each time it times out.

The receiver's first timeout, which sends a <nak>, signals the transmitter to start the transfer of data.

The receiver could send a <nak> immediately, in case the sender is ready. This would work if the sender were a version of XMODEM, which is a variant of MODEM which must be initiated remotely by the receiver.

This would save the initial 10 second timeout, and is used by CMODEM. However, the receiver must continue to timeout every 10 seconds in case the sender wasn't ready.

Once the receiver has begun receiving a block of data it will use a one-second timeout for each character and the checksum. If the receiver wishes to <nak> a block for any reason (invalid header, timeout receiving data), it must wait for the line to clear. See "programming considerations" for various techniques used by different modem programs for handling error conditions.

If a valid block number is received, it will be one of the following:

1) the expected one, in which case everything is fine;

2) a repeat of a previously received block; this should be considered OK, since it indicates that the receiver's <ack> was glitched, and the sender re-transmitted; the only special handling is to discard the received block of data;

3) any other block number will indicate a loss of synchronization, such as the sender getting a line-glitch that appeared as an <ack>; this condition normally is not recoverable, so the receiver normally aborts the transmission by sending a <can>.

SENDING PROGRAM CONSIDERATIONS

While waiting for the transmission to begin, the sender may have only a single long timeout, normally one minute in length, or it may have a shorter timeout, normally 10 seconds in length, before retrying the transmission, for some number (such as 10) of retries. Cmodem uses the latter method, as this aborts fewer transmissions. The use of this 10 second timeout makes the transmission time-sensitive, whereas the one long timeout allows the entire process to be completely receiver-driven. The 10 second timeout is compatible with any of the existing MODEM protocol programs.

When the sender has no more data, it will send an <eot>. It will then wait for an <ack>. The sender will resend the <eot> if an <ack> is not received, just as in normal transmission.

Another procedure, used by some MODEM protocol programs, although not by CMODEM, is to have the sender implement only a high-level one-minute timeout after sending the <eot> before aborting the transmission. This would give the receiving end more control of the data exchange procedure.

DATA FLOW EXAMPLE

To show an example of a data flow session, a 3-block message is diagrammed below. It includes the two most common line hits: a garbaged block, and an <ack> reply getting garbaged. <xx> represents the checksum byte.

```
SENDER                              RECEIVER
                       (times out after 10 seconds)
                   <---            <nak>
<soh> 01 FE -data- <xx>   --->
                   <---            <ack>
<soh> 02 FD -data- xx     ---> (data gets line hit)
                   <---            <nak>
<soh> 02 FD -data- xx     --->
                   <---            <ack>
<soh> 03 FC -data- xx     --->
(ack gets garbaged)       <---            <ack>
<soh> 03 FC -data- xx     --->            <ack>
<eot>                     --->
                   <---            <ack>
```

PROGRAMMING CONSIDERATIONS

Following are suggestions published in various sources for potential MODEM-compatible program writers or modifiers. Most of them are included in the CMODEM program.

The character-receive subroutine should be called with a parameter that specifies the number of seconds to wait. The receiver should first call it with a time of 10, then <nak> and try again, 10 times.

After receiving the <soh>, the receiver should call the character receive subroutine with a 1-second timeout, for the remainder of the message and the <cksum>.

Since they are sent as a continuous stream, timing out of this implies a serious problem that may have been caused by the receipt of only 127 characters instead of 128.

When the receiver wishes to transmit a <nak>, it should call a "PURGE" subroutine, and wait for the line to clear. The sender should have discarded any characters in its internal hardware and software receive buffers immediately upon sending a block, to help prevent misinterpretation of line glitches.

The most common technique is for "PURGE" to call the character receive subroutine, specifying a 1-second timeout, and looping back until a timeout actually occurs.

The receiver then sends the <nak>, hopefully ensuring that the transmitter will receive it properly.

Some may wish to add code to the character receive routine to set an error flag if the ACIA or other device on the modem port shows framing error or overrun. This code is not currently included in CMODEM.

This will help catch a few more glitches, the most common of which is a hit in the high bits of the byte in two consecutive bytes. The checksum is unaffected in this case, since counting in eight bits and discarding the carry bits produces the same results when adding (80H + 80H) as with adding (00H + 00H).

SUMMARY

The preceding has discussed the application of OS/9 to communicating with other systems and with other terminals. The method described, a program named CMODEM, implements several modes of communication (some of which are compatible with CP/M and other systems), which attempt to provide an error-free means of communication despite potentially-noisy telephone lines, and other modes of communications, which, while not error-free, are compatible with other systems and terminals.

POSTSCRIPT

The development of this program has proven to be a real learning experience. This was due to the many pieces of code that had to be written to handle issues that are not a part of average applications programming environment. Much of the program can be considered closer to operating system programming than to applications.

There also tends to be a never ending supply of "one more little feature" requests that keep the program in a constant state of change and also bring out latent bugs that did not seem to exist before.

REVIEW OF SDISK, BOOTFIX, AND FILTER KIT 1

REVIEW OF D P JOHNSON'S
SDISK, BOOTFIX, AND
FILTER KIT 1

by E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-1717/4570

The standard COCO OS/9 disk drivers support only the SSDD 35 track non-standard disk format used by Tandy on the COCO. This is very inconvenient for those users and developers concerned with buying or marketing OS/9 software, as most existing OS/9 software was developed before COCO OS/9 was available. In order to port software to COCO OS/9, it was necessary to send it thru the TI port on the COCO at 300 Baud (and hope that OS/9 did not garbage it too much), or use the FHL O-PAK XCOPY utility (which works, but is tedious for many files), or use the new GIMIX COCO drivers (if you have a GIMIX system, which I do not), or type it in by hand, or ....

D P Johnson has a solution to the incompatible disk format problem which also extends the capabilities of the COCO for those who have drives with more capacity or speed than the SS 35-track drives sold by Tandy. It is called SDISK and costs $29.95 (+$1.00 shipping), but is worth several times that amount in the terms of the convenience it affords the user. It supports 1 to 3 SS or DS, SD or DD, 35 or 40 or 80 track 5.25" drives.

SDISK is composed of an OS/9 device driver (SDISK), several device descriptors (/SDx), a format program (SFORMAT), and a configurator (DESCGEN).

If all that is desired is to access and create standard OS/9 diskettes under COCO OS/9, the user runs DESCGEN once to generate a new device descriptor (/SD0, /SD1, ...) for each drive and saves them, along with a copy of the SDISK module, on a COCO OS/9 diskette. Then, whenever the SDISK and /SDx modules are have been loaded, each drive has a dual identity: /Dx for COCO OS/9 format, and /SDx for standard OS/9 format. All standard OS/9 programs, with the exception of FORMAT, work equally well with either type of device descriptor. The SFORMAT program is used in place of FORMAT; it works with either type of descriptor, so the original FORMAT program could be deleted and replaced by a renamed SFORMAT.

If the user desires to construct a bootable SS diskette already containing SDISK and /SDx, the manual accompanying SDISK provides the detailed directions for accomplishing it. Essentially, the user constructs the device descriptors with DESCGEN and uses OS9GEN to include them in the boot, along with SDISK.

In order to construct a bootable DS diskette (or to construct a single-stage bootable diskette for those systems without the DOS command), Johnson offers the BOOTFIX program for $9.95. It is used on a diskette freshly formatted by SFORMAT and COBBLER to rearrange the bootfile and kernal on track 34 so that the boot will be able to properly access them.

SDISK also provides three extensions to the GETSTT/SETSTT OS/9 functions, as follows:

```
SS.DREAD (GETSTT code $80)
     direct sector read
     with specified sector size,
          side, density, tracks
SS.DWRIT (SETSTT code $80)
     direct sector read
     with specified sector size,
          side, density, tracks
SS.UNFRZ (SETSTT code $81)
     unfreeze DD. information
```

In summary, the SDISK and BOOTFIX package provide the facilities for standard OS/9 disk format operations which should have been provided under COCO OS/9, rather than the restricted form actually present. The implementation is performed in an almost transparent manner, using the normal facilities of OS/9.

The other package from D P Johnson that I reviewed was FILTER KIT 1. It contains the following utility programs, and costs $29.95:

ls — lists file names in current directory, with matching logic
buf — buffers input to eof, then passes it to output
cp — copies files in current directory to files of same names on output path, restricted to list of files on standard input
dl — deletes list of files from current directory
filst — copies contents of list of files from current directory
info — displays detailed file information
mv — moves list of files from current directory to output path
pag — reads standard input, producing a paginated listing
remove — removes file name from directory without releasing space
sell — changes owner numbers of list of file names
setat — changes file attributes of list of file names
sort — performs in-memory sort of records on standard input to standard output

The filter kit may be used on any OS/9 system, and is not restricted to the COCO version of OS/9. The utilities which I tried all seemed to perform as expected. If even one of them is useful, it is probably worth the cost of the entire package.

The address and telephone number for D P Johnson are as follows:

D P Johnson
7655 Cedarcrest Street
Portland, OR 97223
(503) 244-8152

- - -

DISASSEMBLER

CONTINUED FROM LAST MONTH

```
                    *
                    * SEARCHES TAG TABLE FOR ADDRESS IN D REG.
                    * IF FOUND, CARRY IS CLEAR; ELSE IT IS SET.
                    *
0F80 34   06        FINDTG  PSHS  D
0F82 CC   6000              LDD   #$6000     *3 x 2^13
0F85 DD   1E                STD   BIND
0F87 9E   07                LDX   TAGST
0F89 30   1D                LEAX  -3,X
0F8B 30   8B        FINDT1  LEAX  D,X        *BINARY ADJUST
0F8D 9C   09                CMPX  TAGEND     *WITHIN TABLE?
0F8F 24   09                BCC   FINDT2     *NO; GO LOWER
0F91 EC   E4                LDD   0,S        *TAG VALUE
0F93 10A3 01                CMPD  1,X        *TABLE VALUE
0F96 27   11                BEQ   FINDT4     *FOUND TAG
0F98 24   08                BCC   FINDT3     *ADD NEXT OFFSET
0F9A 8D   0F        FINDT2  BSR   FINDT5     *DO BINARY DIVIDE
0F9C 25   08                BCS   FINDT4     *TAG NOT FOUND
0F9E 43                     COMA             *NEGATE FOR SUBTRACT
0F9F 53                     COMB
0FA0 C3   0001              ADDD  #1
0FA3 20   E6                BRA   FINDT1     *NEXT TRY
0FA5 8D   04        FINDT3  BSR   FINDT5     *BINARY DIVIDE
0FA7 24   E2                BCC   FINDT1     *NEXT TRY
0FA9 35   86        FINDT4  PULS  D,PC       *RETURN TO PGM
0FAB 04   1E        FINDT5  LSR   BIND       *BINARY DIVIDE
0FAD 06   1F                ROR   BIND+1     *CARRY SET = DONE
0FAF DC   1E                LDD   BIND
0FB1 39                     RTS
                            PAG
                    * PUTS THE ORG PSEUDO OP IN
                    *THE PRINT LINE.
```

```
                    *
0FB2 8E   1260      DOORG   LDX   #ORGS
0FB5 108E 0069              LDY   #POPCD
0FB9 C6   07                LDB   #7
0FBB A6   80        DOORG1  LDA   0,X+
0FBD A7   A0                STA   0,Y+
0FBF 5A                     DECB
0FC0 26   F9                BNE   DOORG1
0FC2 96   19                LDA   NXTOP
0FC4 17   FE94              LBSR  HEXASC
0FC7 ED   A1                STD   0,Y++
0FC9 96   1A                LDA   NXTOP+1
0FCB 17   FE8D              LBSR  HEXASC
0FCE ED   A4                STD   0,Y
0FD0 8D   40                BSR   PRINT
0FD2 8D   31                BSR   CLRPLN
0FD4 39                     RTS
                            PAG
                    *
                    * WRITES THE TAG, OPCODE, & OPERAND
                    * TO DISC, SUPRESSING EXTRA SPACES ($A0).
                    *
0FD5 8E   0259      WRTDSC  LDX   #FCBOUT    *WRITE SOURCE TO DISC
0FD8 108E 0063              LDY   #PTAG      *STARTING AT TAG
0FDC 108C 0088      WRTDS1  CMPY  #PLINE+62
0FE0 27   1D                BEQ   WRTDS3     *DONE
0FE2 A6   A0                LDA   0,Y+
0FE4 84   7F        WRTDS5  ANDA  #$7F
0FE6 BD   D406              JSR   FMS
0FE9 26   19                BNE   WRTDS4     *ERROR
0FEB A6   3F                LDA   -1,Y       *GET IT BACK
0FED 81   A0                CMPA  #$A0
0FEF 26   EB                BNE   WRTDS1
0FF1 108C 0088      WRTDS2  CMPY  #PLINE+62
0FF5 27   08                BEQ   WRTDS3
0FF7 A6   A0                LDA   0,Y+       *SQUISH SPACES
0FF9 81   A0                CMPA  #$A0       *SPECIAL SPACE
0FFB 26   E7                BNE   WRTDS5
0FFD 20   F2                BRA   WRTDS2
0FFF 86   0D        WRTDS3  LDA   #$D        *C/R
1001 BD   D406              JSR   FMS
1004 39             WRTDS4  RTS              *TEST ERROR ELSEWHERE
                            PAG
1005 8E   004A      CLRPLN  LDX   #PLINE     *CLEAR PRINTLINE
1008 86   A0                LDA   #$A0
100A C6   4F                LDB   #79
100C A7   80        CLRPL1  STA   0,X+
100E 5A                     DECB
100F 26   FB                BNE   CLRPL1
1011 39                     RTS
                    *
                    * PRINTS THE PRINTLINE.
                    *
1012 96   13        PRINT   LDA   LINENO
1014 17   FE44              LBSR  HEXASC
1017 DD   4A                STD   PLNO
1019 96   14                LDA   LINENO+1
101B 17   FE3D              LBSR  HEXASC
101E DD   4C                STD   PLNO+2
1020 96   14                LDA   LINENO+1   *BUMP THE LINE #
1022 8B   01                ADDA  #1
1024 19                     DAA
1025 97   14                STA   LINENO+1
1027 96   13                LDA   LINENO
1029 89   00                ADCA  #0
102B 19                     DAA
102C 97   13                STA   LINENO
102E 8E   004A              LDX   #PLNO
1031 8D   5E                BSR   ZSUPP      *ZERO SUPPRESS
1033 B6   CC03              LDA   DEPTH
1036 27   09                BEQ   PRINT3     *NO HEADINGS
1038 80   04                SUBA  #4
103A B1   CC1A              CMPA  CURLN
103D 26   02                BNE   PRINT3     *NOT TOP PAGE
103F 8D   3A                BSR   PAGE       *PRINT HEADING
1041 8E   004A      PRINT3  LDX   #PLINE
1044 BD   CD1E              JSR   PSTRNG     *PRINT IT
1047 8D   8C                BSR   WRTDSC     *ALL LINES GO TO DISC
1049 39                     RTS
                    *
104A FC   0117      PHEAD   LDD   HPGNO      *PRINT HEADING
104D 1E   89                EXG   A,B
104F 8B   01                ADDA  #1
1051 19                     DAA
1052 1E   89                EXG   A,B
1054 89   00                ADCA  #0
1056 19                     DAA
1057 FD   0117              STD   HPGNO
105A 17   FDFE              LBSR  HEXASC
105D FD   0112              STD   HPAGE
1060 B6   0118              LDA   HPGNO+1
1063 17   FDF5              LBSR  HEXASC
1066 FD   0114              STD   HPAGE+2
1069 8E   0112              LDX   #HPAGE
106C 8D   23                BSR   ZSUPP      *ZERO SUPPRESS
106E 8D   1A                BSR   FOOT1
1070 8D   18                BSR   FOOT1
1072 8E   0003              LDX   #HEAD
1075 BD   CD1E              JSR   PSTRNG
1078 8D   10                BSR   FOOT1
107A 39                     RTS
                    *
107B B6   CC1A      PAGE    LDA   CURLN      *PRINT FOOTER (SPACES)
107E B1   CC03              CMPA  DEPTH
1081 27   04                BEQ   PAGE1
1083 8D   05                BSR   FOOT1
1085 20   F4                BRA   PAGE
```

```
1087 8D    C1        PAGE1   BSR    PHEAD
1089 39                      RTS
                     *
108A 8E    0099      FOOT1   LDX    #HEAD4
108D BD    CD1E              JSR    PSTRNG
1090 39                      RTS
                     *
1091 A6    80        ZSUPP   LDA    0,X+        *ZERO SUPRESS
1093 81    30                CMPA   #'0
1095 26    06                BNE    ZSUPP1
1097 86    20                LDA    #$20
1099 A7    1F                STA    -1,X
109B 20    F4                BRA    ZSUPP
109D 39              ZSUPP1  RTS
                             PAG
                     *SORTS XRTAG INTO THE CROSS REFERENCE
                     *TABLE.
                     *
109E 34    16        XRSORT  PSHS   X,D
10A0 9E    44                LDX    XREND
10A2 30    05                LEAX   5,X
10A4 BC    CC28              CMPX   MEMEND
10A7 25    06                BCS    XRSOR1
10A9 30    1B                LEAX   -5,X
10AB 9F    42                STX    XRST        *BEYOND MEMORY
10AD 20    75                BRA    XRSOR9
10AF CC    5000      XRSOR1  LDD    #$5000      * 5 * 2^12
10B2 DD    1E                STD    BIND
10B4 9E    42                LDX    XRST
10B6 30    1B                LEAX   -5,X
10B8 30    8B        XRFIN1  LEAX   D,X
10BA 9C    44                CMPX   XREND
10BC 24    09                BCC    XRFIN2      *LOWER
10BE DC    47                LDD    XRTAG+1
10C0 10A3  01                CMPD   1,X
10C3 27    13                BEQ    XRFIN4      *FOUND
10C5 24    0C                BCC    XRFIN3      *ADD
10C7 17    FEE1      XRFIN2  LBSR   FINDT5      *SUBTRACT
10CA 25    0C                BCS    XRFIN4      *NOT THERE
10CC 43                      COMA
10CD 53                      COMB
10CE C3    0001              ADDD   #1
10D1 20    E5                BRA    XRFIN1
10D3 17    FED5      XRFIN3  LBSR   FINDT5
10D6 24    E0                BCC    XRFIN1
10D8 9C    44        XRFIN4  CMPX   XREND
10DA 27    0B                BEQ    XRFIN6      *GOES HERE
10DC DC    47                LDD    XRTAG+1
10DE 10A3  01                CMPD   1,X
10E1 25    04                BCS    XRFIN6      *AT OR BEYOND
10E3 30    05                LEAX   5,X
10E5 20    F1                BRA    XRFIN4
10E7 DC    46        XRFIN6  LDD    XRTAG
10E9 2A    11                BPL    XRFIN8      *GOES HERE
10EB 9C    42        XRFIN7  CMPX   XRST
10ED 27    0D                BEQ    XRFIN8      *GOES HERE
10EF DC    47                LDD    XRTAG+1
10F1 10A3  1C                CMPD   -4,X
10F4 25    2                 BCS    XRFI10
10F6 26    04                BNE    XRFIN8      *GOES HERE
10F8 30    1B        XRFI10  LEAX   -5,X
10FA 20    EF                BRA    XRFIN7
10FC 9F    1E        XRFIN8  STX    BIND        *TEMP STORE
10FE 9E    44                LDX    XREND
1100 30    05                LEAX   5,X
1102 9F    44                STX    XREND
1104 30    1B        XRFIN9  LEAX   -5,X
1106 9C    1E                CMPX   BIND
1108 27    0E                BEQ    XRFI11
110A A6    1B                LDA    -5,X
110C A7    84                STA    0,X
110E EC    1C                LDD    -4,X
1110 ED    01                STD    1,X
1112 EC    1E                LDD    -2,X
1114 ED    03                STD    3,X
1116 20    EC                BRA    XRFIN9
1118 96    46        XRFI11  LDA    XRTAG
111A A7    84                STA    0,X
111C DC    47                LDD    XRTAG+1
111E ED    01                STD    1,X
1120 DC    13                LDD    LINENO
1122 ED    03                STD    3,X
1124 35    96        XRSOR9  PULS   X,D,PC
                             PAG
                     *
                     *PRINTS THE CROSS REFERENCE
                     *
1126 B6    CC03      XRPRNT  LDA    DEPTH
1129 27    03                BEQ    XRPRN6
112B 17    FF4D              LBSR   PAGE
112E 9E    42        XRPRN6  LDX    XRST
1130 9C    44        XRPRN1  CMPX   XREND
1132 27    5B                BEQ    XRPRN5      *DONE
1134 34    10                PSHS   X
1136 B6    CC03              LDA    DEPTH
1139 27    0A                BEQ    XRPRN7
113B 80    04                SUBA   #4
113D B1    CC1A              CMPA   CURLN
1140 26    03                BNE    XRPRN7

1142 17    FF36              LBSR   PAGE
1145 BD    CD24      XRPRN7  JSR    PCRLF
1148 35    10                PULS   X
114A 10AE  01                LDY    1,X
114D C6    0E                LDB    #14         *MAX ADDR'S IN LINE
114F 8D    42                BSR    XRPLN       *PRINT LINE & SPACE
1151 34    34                PSHS   X,Y,B
```

```
1153 A6    84                LDA    0,X
1155 BD    CD18              JSR    PUTCHR
1158 AE    61                LDX    1,S
115A 30    01                LEAX   1,X
115C BD    CD45              JSR    OUTADR      *REST OF TAG
115F 86    20                LDA    #$20
1161 BD    CD18              JSR    PUTCHR
1164 86    20                LDA    #$20
1166 BD    CD18              JSR    PUTCHR
1169 86    20                LDA    #$20
116B BD    CD18              JSR    PUTCHR
116E 35    34                PULS   X,Y,B
1170 30    05        XRPRN2  LEAX   5,X
1172 10AC  01                CMPY   1,X
1175 26    B9                BNE    XRPRN1      *NEW TAG
1177 5D                      TSTB
1178 26    11                BNE    XRPRN4
117A C6    0D                LDB    #13
117C 34    34                PSHS   X,Y,B
117E BD    CD24              JSR    PCRLF
1181 86    20        XRPRN3  LDA    #$20
1183 BD    CD18              JSR    PUTCHR      *OUTPUT 13 SPACES
1186 5A                      DECB
1187 26    F8                BNE    XRPRN3
1189 35    34                PULS   B,X,Y
118B 8D    06        XRPRN4  BSR    XRPLN       *PRINT TAG LINE & SPACE
118D 20    E1                BRA    XRPRN2
118F BD    CD24      XRPRN5  JSR    PCRLF
1192 39                      RTS
                     *
1193 34    34        XRPLN   PSHS   X,Y,B
1195 A6    03                LDA    3,X
1197 17    FCC1              LBSR   HEXASC
119A DD    94                STD    HEAD4-5
119C A6    04                LDA    4,X
119E 17    F BA              LBSR   HEXASC
11A1 DD    96                STD    HEAD4-3
11A3 86    20                LDA    #$20
11A5 97    98                STA    HEAD4-1
11A7 8E    0094              LDX    #HEAD4-5  *ZERO SUPRESS
11AA A6    80        XRPLN1  LDA    0,X+
11AC 81    30                CMPA   #'0
11AE 26    06                BNE    XRPLN2
11B0 86    20                LDA    #$20
11B2 A7    1F                STA    -1,X
11B4 20    F4                BRA    XRPLN1
11B6 C6    05        XRPLN2  LDB    #5
11B8 8E    0094              LDX    #HEAD4-5
11BB A6    80        XRPLN4  LDA    0,X+
11BD BD    CD18              JSR    PUTCHR
11C0 5A                      DECB
11C1 26    F8                BNE    XRPLN4
11C3 6A    E4                DEC    0,S
11C5 35    B4                PULS   X,Y,B,PC
                             PAG
11C7 4E 4F 54 20     MSG2    FCC    "NOT ENOUGH MEMORY; ABORTING."
11CB 45 4E 4F 55
11CF 47 48 20 4D
11D3 45 4D 4F 52
11D7 59 3B 20 41
11DB 42 4F 52 54
11DF 49 4E 47 2E
11E3 04                      FCB    4
11E4 42 49 4E 41     MSG4    FCC    "BINARY FORMAT ERROR"
11E8 52 59 20 46
11EC 4F 52 4D 41
11F0 54 20 45 52
11F4 52 4F 52
11F7 04                      FCB    4
11F8 A0              TFXG    FCB    $A0         *TRANSFER/EXCHANGE OPNDS
11F9 44                      FCC    "D"
11FA A0                      FCB    $A0
11FB 58                      FCC    "X"
11FC A0                      FCB    $A0
11FD 59                      FCC    "Y"
11FE A0                      FCB    $A0
11FF 55                      FCC    "U"
1200 A0                      FCB    $A0
1201 53 50 43                FCC    "SPC"
1204 A0                      FCB    $A0
1205 41                      FCC    "A"
1206 A0                      FCB    $A0
1207 42 43 43 44             FCC    "BCCDP"
120B 50
120C 43 43           PSPL    FCC    "CC"        *PUSH/PULL OPERANDS
120E A0                      FCB    $A0
120F 41                      FCC    "A"
1210 A0                      FCB    $A0
1211 42 44 50                FCC    "BDP"
1214 A0                      FCB    $A0
1215 58                      FCC    "X"
1216 A0                      FCB    $A0
1217 59 53 55 50             FCC    "YSUPC"
121B 43
121C 44 50 20 4C     DPMSG   FCC    "DP LOADED"
1220 4F 41 44 45
1224 44
1225 44 49 52 45     DPMSG1  FCC    "DIRECT ADDR"
1229 43 54 20 41
122D 44 44 52
1230 2A 57 41 52     DPMSG2  FCC    "*WARNING: DIRECT PAGE ADDRESSING "
1234 4E 49 4E 47
1238 3A 20 44 49
123C 52 45 43 54
1240 20 50 41 47
1244 45 20 41 44
```

```
1248 44 52 45 53
124C 53 49 4E 47
1250 20
1251 55 53 45 44            FCC    "USED!"
1255 21
1256 53 45 54 44    OPMSG3  FCC    "SETUP 0"
125A 50 20 30
125D 4E 41 4D       NAMS    FCC    "NAM"
1260 4F 52 47       ORGS    FCC    "ORG"
1263 A0 A0 A0               F B    $A0,$A0,$A0
1266 24                     FCC    "$"
1267 45 51 55       EQUS    FCC    "EQU"
126A 46 43 42       FCBS    FCC    "FCB"
126D A0 A0 A0               FCB    $A0,$A0,$A0
1270 24                     FCC    "$"
1271 45 4E 44       ENDS    FCC    "END"
1274 46 43 43       FCCS    FCC    "FCC"
1277 A0 A0 A0               FCB    $A0,$A0,$A0
127A 22                     FCB    $22
127B 4F 50 54       OPTS    FCC    "OPT"
127E A0 A0 A0               F B    $A0,$A0,$A0
                    *END OF SUPPORT MODULES
1281 50 41 47               FCC    "PAG"
                    * LIB 1.DISTBL.TXT
1284 2122 2324      LOOK10  FDB    $2122,$2324,$2526,$2728,$292A
1288 2526 2728
128C 292A
128E 2B2C 2D2E              FDB    $2B2C,$2D2E,$2F3F,$838C,$8E93,$9C9E
1292 2F3F 838C
1296 8E93 9C9E
129A 9FA3 ACAE              FDB    $9FA3,$ACAE,$AFB3,$BCBE,$BFCE,$DEDF
129E AFB3 BCBE
12A2 BFCE DEDF
12A6 EE EF FE FF            FCB    $EE,$EF,$FE,$FF
               12AA LK10E   EQU    *          *DON'T SCREW WITH THIS TAG
12AA 3F83 8C93      LOOK11  FDB    $3F83,$8C93,$9CA3,$ACB3
12AE 9CA3 ACB3
12B2 BC                     FCB    $BC
               12B3 LK11E   EQU    *          *DON'T MISTREAT EITHER!
                    ILLEG   MACRO
                            FCB    $FF
                            RMB    6
                            ENDM
                    *END NOTHING START 0x
               12B3 ZZZZZ   EQU    *
12B3 4E 45 47       CMDTBL  FCC    "NEG"      *00
12B6 A0 A0                  FCB    $A0,$A0    *SPECIAL SPACE
12B8 2288                   FDB    $2288
12BA                        ILLEG
12C1                        ILLEG
12C8 43 4F 4D               FCC    "COM"      * 03
12CB A0 A0                  FCB    $A0,$A0
12CD 2288                   FDB    $2288
12CF 4C 53 52               FCC    "LSR"      * 04
12D2 A0 A0                  FCB    $A0,$A0
12D4 2288                   FDB    $2288
12D6                        ILLEG
12DD 52 4F 52               FCC    "ROR"      * 06
12E0 A0 A0                  FCB    $A0,$A0
12E2 2288                   FDB    $2288
12E4 41 53 52               FCC    "ASR"      *07
12E7 A0 A0                  FCB    $A0,$A0
12E9 2288                   FDB    $2288
12EB 41 53 4C               FCC    "ASL"      * 08
12EE A0 A0                  FCB    $A0,$A0
12F0 2288                   FDB    $2288
12F2 52 4F 4C               FCC    "ROL"      *09
12F5 A0 A0                  FCB    $A0,$A0
12F7 2288                   FDB    $2288
12F9 44 45 43               FCC    "DEC"      * 0A
12FC A0 A0                  FCB    $A0,$A0
12FE 2288                   FDB    $2288
1300                        ILLEG
1307 49 4E 43               FCC    "INC"      *0C
130A A0 A0                  FCB    $A0,$A0
130C 2288                   FDB    $2288
130E 54 53 54               FCC    "TST"      * 0D
1311 A0 A0                  FCB    $A0,$A0
1313 2288                   FDB    $2288
1315 4A 4D 50               FCC    "JMP"      * 0E
1318 A0 A0                  FCB    $A0,$A0
131A AA88                   FDB    $AA88
131C 43 4C 52               FCC    "CLR"      * 0F
131F A0 A0                  FCB    $A0,$A0
1321 2288                   FDB    $2288
                    *END 0x START 1x
132310                      FCB    $10
1324                        RMB    6
132A 11                     FCB    $11
132B                        RMB    6
1331 4E 4F 50               FCC    "NOP"      *12
1334 A0 A0                  FCB    $A0,$A0
1336 0180                   FDB    $0180
1338 53 59 4E 43            FCC    "SYNC"     *13
133C A0                     FCB    $A0
133D 0180                   FDB    $0180
133F                        ILLEG
1346                        ILLEG
134D 4C 42 52 41            FCC    "LBRA"     *16
1351 A0                     FCB    $A0
1352 DB8C                   FDB    $DB8C
1354 4C 42 53 52            FCC    "LBSR"
1358 A0                     FCB    $A0
1359 DB8C                   FDB    $DB8C
135B                        ILLEG
1362 44 41 41               FCC    "DAA"
```

```
1365 A0 A0                  F B    $A0,$A0
1367 0180                   FDB    $0180
1369 4F 52 43 43            FCC    "ORCC"
136D A0                     FCB    $A0
136E 4288                   FDB    $4288
1370                        ILLEG
1377 41 4E 44 43            FCC    "ANDCC"
137B 43
137C 4288                   FDB    $4288
137E 53 45 58               FCC    "SEX"
1381 A0 A0                  FCB    $A0,$A0
1383 0180                   FDB    $0180
1385 45 58 47               FCC    "EXG"
1388 A0 A0                  FCB    $A0,$A0
138A 0288                   FDB    $0288
138C 54 46 52               FCC    "TFR"
138F A0 A0                  FCB    $A0,$A0
1391 0288                   FDB    $0288
                    *END 1x START 2x
1393 42 52 41               FCC    "BRA"
1396 A0 A0                  FCB    $A0,$A0
1398 DA88                   FDB    $DA88
139A 42 52 4E               FCC    "BRN"
139D A0 A0                  FCB    $A0,$A0
139F D288                   FDB    $D288
13A1 42 48 49               FCC    "BHI"
13A4 A0 A0                  FCB    $A0,$A0
13A6 D288                   FDB    $D288
13A8 42 4C 53               FCC    "BLS"
13AB A0 A0                  FCB    $A0,$A0
13AD D288                   FDB    $D288
13AF 42 43 43               FCC    "BCC"
13B2 A0 A0                  FCB    $A0,$A0
13B4 D288                   FDB    $D288
13B6 42 43 53               FCC    "BCS"
13B9 A0 A0                  F B    $A0,$A0
13BB D288                   FDB    $D288
13BD 42 4E 45               FCC    "BNE"
13C0 A0 A0                  FCB    $A0,$A0
13C2 D288                   FDB    $D288
13C4 42 45 51               FCC    "BEQ"
13C7 A0 A0                  FCB    $A0,$A0
13C9 D288                   FDB    $D288
13CB 42 56 43               FCC    "BVC"
13CE A0 A0                  FCB    $A0,$A0
13D0 D288                   FDB    $D288
13D2 42 56 53               FCC    "BVS"
13D5 A0 A0                  FCB    $A0,$A0
13D7 D288                   FDB    $D288
13D9 42 50 4C               FCC    "BPL"
13DC A0 A0                  FCB    $A0,$A0
13DE D288                   FDB    $D288
13E0 42 4D 49               FCC    "BMI"
13E3 A0 A0                  FCB    $A0,$A0
13E5 D288                   FDB    $D288
13E7 42 47 45               FCC    "BGE"
13EA A0 A0                  FCB    $A0,$A0
13EC D288                   FDB    $D288
13EE 42 4C 54               FCC    "BLT"
13F1 A0 A0                  FCB    $A0,$A0
13F3 D288                   FDB    $D288
13F5 42 47 54               FCC    "BGT"
13F8 A0 A0                  FCB    $A0,$A0
13FA D288                   FDB    $D288
13F  42 4C 45               FCC    "BLE"
13FF A0 A0                  FCB    $A0,$A0
1401 D288                   FDB    $D288
                    *END 2x START 3x
1403 4C 45 41 58            FCC    "LEAX"
1407 A0                     FCB    $A0
1408 1288                   FDB    $1288
140A 4C 45 41 59            FCC    "LEAY"
140E A0                     FCB    $A0
140F 1288                   FDB    $1288
1411 4C 45 41 53            FCC    "LEAS"
1415 A0                     FCB    $A0
1416 1288                   FDB    $1288
1418 4C 45 41 55            FCC    "LEAU"
141C A0                     FCB    $A0
141D 1288                   FDB    $1288
141F 50 53 48 53            FCC    "PSHS"
1423 A0                     FCB    $A0
1424 0288                   FDB    $0288
1426 50 55 4C 53            FCC    "PULS"
142A A0                     FCB    $A0
142B 0288                   FDB    $0288
142D 50 53 48 55            FCC    "PSHU"
1431 A0                     FCB    $A0
1432 0288                   FDB    $0288
1434 50 55 4C 55            FCC    "PULU"
1438 A0                     FCB    $A0
1439 0288                   FDB    $0288
143B                        ILLEG
1442 52 54 53               FCC    "RTS"
1445 A0 A0                  FCB    $A0,$A0
1447 0980                   FDB    $0980
1449 41 42 58               FCC    "ABX"
144C A0 A0                  FCB    $A0,$A0
144E 0180                   FDB    $0180
1450 52 54 49               FCC    "RTI"
1453 A0 A0                  FCB    $A0,$A0
1455 0980                   FDB    $0980
1457 43 57 41 49            FCC    "CWAI"
145B A0                     FCB    $A0
145C 4288                   FDB    $4288
145E 4D 55 4C               FCC    "MUL"
```

```
1461 A0 A0            FCB   $A0,$A0
1463 0180             FDB   $0180
1465                  ILLEG
146C 53 57 49         FCC   "SWI"
146F A0 A0            FCB   $A0,$A0
1471 0180             FDB   $0180
             *END 3x  START 4x
1473 4E 45 47 41      FCC   "NEGA"
1477 A0               FCB   $A0
1478 0180             FDB   $0180
147A                  ILLEG
1481                  ILLEG
1488 43 4F 4D 41      FCC   "COMA"
148C A0               FCB   $A0
148D 0180             FDB   $0180
148F 4C 53 52 41      FCC   "LSRA"
1493 A0               FCB   $A0
1494 0180             FDB   $0180
1496                  ILLEG
149D 52 4F 52 41      FCC   "RORA"
14A1 A0               FCB   $A0
14A2 0180             FDB   $0180
14A4 41 53 52 41      FCC   "ASRA"
14A8 A0               FCB   $A0
14A9 0180             FDB   $0180
14AB 41 53 4C 41      FCC   "ASLA"
14AF A0               FCB   $A0
14B0 0180             FDB   $0180
14B2 52 4F 4C 41      FCC   "ROLA"
14B6 A0               FCB   $A0
14B7 0180             FDB   $0180
14B9 44 45 43 41      FCC   "DECA"
14BD A0               FCB   $A0
14BE 0180             FDB   $0180
14C0                  ILLEG
14C7 49 4E 43 41      FCC   "INCA"
14CB A0               FCB   $A0
14CC 0180             FDB   $0180
14CE 54 53 54 41      FCC   "TSTA"
14D2 A0               FCB   $A0
14D3 0180             FDB   $0180
14D5                  ILLEG
14DC 43 4C 52 41      FCC   "CLRA"
14E0 A0               FCB   $A0
14E1 0180             FDB   $0180
             *END 4x  START 5x
14E3 4E 45 47 42      FCC   "NEGB"
14E7 A0               FCB   $A0
14E8 0180             FDB   $0180
14EA                  ILLEG
14F1                  ILLEG
14F8 43 4F 4D 42      FCC   "COMB"
14FC A0               FCB   $A0
14FD 0180             FDB   $0180
14FF 4C 53 52 42      FCC   "LSRB"
1503 A0               FCB   $A0
1504 0180             FDB   $0180
1506                  ILLEG
150D 52 4F 52 42      FCC   "RORB"
1511 A0               FCB   $A0
1512 0180             FDB   $0180
1514 41 53 52 42      FCC   "ASRB"
1518 A0               FCB   $A0
1519 0180             FDB   $0180
151B 41 53 4C 42      FCC   "ASLB"
151F A0               FCB   $A0
1520 0180             FDB   $0180
1522 52 4F 4C 42      FCC   "ROLB"
1526 A0               FCB   $A0

1527 0180             FDB   $0180
1529 44 45 43 42      FCC   "DECB"
152D A0               FCB   $A0
152E 0180             FDB   $0180
1530                  ILLEG
1537 49 4E 43 42      FCC   "INCB"
153B A0               FCB   $A0
153C 0180             FDB   $0180
153E 54 53 54 42      FCC   "TSTB"
1542 A0               FCB   $A0
1543 0180             FDB   $0180
1545                  ILLEG
154C 43 4C 52 42      FCC   "CLRB"
1550 A0               FCB   $A0
1551 0180             FDB   $0180
             *END 5x  START 6x
1553 4E 45 47         FCC   "NEG"
1556 A0 A0            FCB   $A0,$A0
1558 1288             FDB   $1288
155A                  ILLEG
1561                  ILLEG
1568 43 4F 4D         FCC   "COM"
156B A0 A0            FCB   $A0,$A0
156D 1288             FDB   $1288
156F 4C 53 52         FCC   "LSR"
1572 A0 A0            FCB   $A0,$A0
1574 1288             FDB   $1288
1576                  ILLEG
157D 52 4F 52         FCC   "ROR"
1580 A0 A0            FCB   $A0,$A0
1582 1288             FDB   $1288
1584 41 53 52         FCC   "ASR"
1587 A0 A0            FCB   $A0,$A0
1589 1288             FDB   $1288
158B 41 53 4C         FCC   "ASL"
158E A0 A0            FCB   $A0,$A0
1590 1288             FDB   $1288
```

```
1592 52 4F 4C         FCC   "ROL"
1595 A0 A0            FCB   $A0,$A0
1597 1288             FDB   $1288
1599 44 45 43         FCC   "DEC"
159C A0 A0            FCB   $A0,$A0
159E 1288             FDB   $1288
15A0                  ILLEG
15A7 49 4E 43         FCC   "INC"
15AA A0 A0            FCB   $A0,$A0
15AC 1288             FDB   $1288
15AE 54 53 54         FCC   "TST"
15B1 A0 A0            FCB   $A0,$A0
15B3 1288             FDB   $1288
15B5 4A 4D 50         FCC   "JMP"
15B8 A0 A0            FCB   $A0,$A0
15BA 9A88             FDB   $9A88
15BC 43 4C 52         FCC   "CLR"
15BF A0 A0            FCB   $A0,$A0
15C1 1288             FDB   $1288
             *END 6x  START 7x
15C3 4E 45 47         FCC   "NEG"
15C6 A0 A0            FCB   $A0,$A0
15C8 338C             FDB   $338C
15CA                  ILLEG
15D1                  ILLEG
15D8 43 4F 4D         FCC   "COM"
15DB A0 A0            FCB   $A0,$A0
15DD 338C             FDB   $338C
15DF 4C 53 52         FCC   "LSR"
15E2 A0 A0            FCB   $A0,$A0
15E4 338C             FDB   $338C
15E6                  ILLEG
15ED 52 4F 52         FCC   "ROR"
15F0 A0 A0            FCB   $A0,$A0
15F2 338C             FDB   $338C
15F4 41 53 52         FCC   "ASR"
15F7 A0 A0            FCB   $A0,$A0
15F9 338C             FDB   $338C
15FB 41 53 4C         FCC   "ASL"
15FE A0 A0            FCB   $A0,$A0
1600 338C             FDB   $338C
1602 52 4F 4C         FCC   "ROL"
1605 A0 A0            FCB   $A0,$A0
1607 338C             FDB   $338C
1609 44 45 43         FCC   "DEC"
160C A0 A0            FCB   $A0,$A0
160E 338C             FDB   $338C
1610                  ILLEG
1617 49 4E 43         FCC   "INC"
161A A0 A0            FCB   $A0,$A0
161C 338C             FDB   $338C
161E 54 53 54         FCC   "TST"
1621 A0 A0            FCB   $A0,$A0
1623 338C             FDB   $338C
1625 4A 4D 50         FCC   "JMP"
1628 A0 A0            FCB   $A0,$A0
162A 888C             FDB   $888C
162C 43 4C 52         FCC   "CLR"
162F A0 A0            FCB   $A0,$A0
1631 338C             FDB   $338C
             *END 7x  START 8x
1633 53 55 42 41      FCC   "SUBA"
1637 A0               FCB   $A0
1638 4288             FDB   $4288
163A 43 4D 50 41      FCC   "CMPA"
163E A0               FCB   $A0
163F 4288             FDB   $4288
1641 53 42 43 41      FCC   "SBCA"
1645 A0               FCB   $A0
1646 4288             FDB   $4288
1648 53 55 42 44      FCC   "SUBD"
164C A0               FCB   $A0
164D 438C             FDB   $438C
164F 41 4E 44 41      FCC   "ANDA"
1653 A0               FCB   $A0
1654 4288             FDB   $4288
1656 42 49 54 41      FCC   "BITA"
165A A0               FCB   $A0
165B 4288             FDB   $4288
165D 4C 44 41         FCC   "LDA"
1660 A0 A0            FCB   $A0,$A0
1662 4288             FDB   $4288
1664                  ILLEG
166B 45 4F 52 41      FCC   "EORA"
166F A0               FCB   $A0
1670 4288             FDB   $4288
1672 41 44 43 41      FCC   "ADCA"
1676 A0               FCB   $A0
1677 4288             FDB   $4288
1679 4F 52 41         FCC   "ORA"
167C A0 A0            FCB   $A0,$A0
167E 4288             FDB   $4288
1680 41 44 44 41      FCC   "ADDA"
1684 A0               FCB   $A0
1685 4288             FDB   $4288
1687 43 4D 50 58      FCC   "CMPX"
168B A0               FCB   $A0

168C 438C             FDB   $438C
168E 42 53 52         FCC   "BSR"
1691 A0 A0            FCB   $A0,$A0
1693 DA88             FDB   $DA88
1695 4C 44 58         FCC   "LDX"
1698 A0 A0            FCB   $A0,$A0
169A 438C             FDB   $438C
169C                  ILLEG
             *END 8x  START 9x
```

```
16A3 53 55 42 41      FCC   "SUBA"
16A7 A0               FCB   $A0
16A8 2288             FDB   $2288
16AA 43 4D 50 41      FCC   "CMPA"
16AE A0               FCB   $A0
16AF 2288             FDB   $2288
16B1 53 42 43 41      FCC   "SBCA"
16B5 A0               FCB   $A0
16B6 2288             FDB   $2288
16B8 53 55 42 44      FCC   "SUBD"
16BC A0               FCB   $A0
16BD 2288             FDB   $2288
16BF 41 4E 44 41      FCC   "ANDA"
16C3 A0               FCB   $A0
16C4 2288             FDB   $2288
16C6 42 49 54 41      FCC   "BITA"
16CA A0               FCB   $A0
16CB 2288             FDB   $2288
16CD 4C 44 41         FCC   "LDA"
16D0 A0 A0            FCB   $A0,$A0
16D2 2288             FDB   $2288
16D4 53 54 41         FCC   "STA"
16D7 A0 A0            FCB   $A0,$A0
16D9 2288             FDB   $2288
16DB 45 4F 52 41      FCC   "EORA"
16DF A0               FCB   $A0
16E0 2288             FDB   $2288
16E2 41 44 43 41      FCC   "ADCA"
16E6 A0               FCB   $A0
16E7 2288             FDB   $2288
16E9 4F 52 41         FCC   "ORA"
16EC A0 A0            FCB   $A0,$A0
16EE 2288             FDB   $2288
16F0 41 44 44 41      FCC   "ADDA"
16F4 A0               FCB   $A0
16F5 2288             FDB   $2288
16F7 43 4D 50 58      FCC   "CMPX"
16FB A0               FCB   $A0
16FC 2288             FDB   $2288
16FE 4A 53 52         FCC   "JSR"
1701 A0 A0            FCB   $A0,$A0
1703 AA88             FDB   $AA88
1705 4C 44 58         FCC   "LDX"
1708 A0 A0            FCB   $A0,$A0
170A 2288             FDB   $2288
170C 53 54 58         FCC   "STX"
170F A0 A0            FCB   $A0,$A0
1711 2288             FDB   $2288
             *END 9x  START Ax
1713 53 55 42 41      FCC   "SUBA"
1717 A0               FCB   $A0
1718 1288             FDB   $1288
171A 43 4D 50 41      FCC   "CMPA"
171E A0               FCB   $A0
171F 1288             FDB   $1288
1721 53 42 43 41      FCC   "SBCA"
1725 A0               FCB   $A0
1726 1288             FDB   $1288
1728 53 55 42 44      FCC   "SUBD"
172C A0               FCB   $A0
172D 1288             FDB   $1288
172F 41 4E 44 41      FCC   "ANDA"
1733 A0               FCB   $A0
1734 1288             FDB   $1288
1736 42 49 54 41      FCC   "BITA"
173A A0               FCB   $A0
173B 1288             FDB   $1288
173D 4C 44 41         FCC   "LDA"
1740 A0 A0            FCB   $A0,$A0
1742 1288             FDB   $1288
1744 53 54 41         FCC   "STA"
1747 A0 A0            FCB   $A0,$A0
1749 1288             FDB   $1288
174B 45 4F 52 41      FCC   "EORA"
174F A0               FCB   $A0
1750 1288             FDB   $1288
1752 41 44 43 41      FCC   "ADCA"
1756 A0               FCB   $A0
1757 1288             FDB   $1288
1759 4F 52 41         FCC   "ORA"
175C A0 A0            FCB   $A0,$A0
175E 1288             FDB   $1288
1760 41 44 44 41      FCC   "ADDA"
1764 A0               FCB   $A0
1765 1288             FDB   $1288
1767 43 4D 50 58      FCC   "CMPX"
176B A0               FCB   $A0
176C 1288             FDB   $1288
176E 4A 53 52         FCC   "JSR"
1771 A0 A0            FCB   $A0,$A0
1773 9A88             FDB   $9A88
1775 4C 44 58         FCC   "LDX"
1778 A0 A0            FCB   $A0,$A0
177A 1288             FDB   $1288
177C 53 54 58         FCC   "STX"
177F A0 A0            FCB   $A0,$A0
1781 1288             FDB   $1288
             *END Ax  START Bx
1783 53 55 42 41      FCC   "SUBA"
1787 A0               FCB   $A0
1788 338C             FDB   $338C
178A 43 4D 50 41      FCC   "CMPA"
178E A0               FCB   $A0
178F 338C             FDB   $338C
1791 53 42 43 41      FCC   "SBCA"
1795 A0               FCB   $A0
1796 338C             FDB   $338C
```

```
1798 53 55 42 44             FCC    "SUBD"
179C A0                      FCB    $A0
179D 338C                    FDB    $338C
179F 41 4E 44 41             FCC    "ANDA"
17A3 A0                      FCB    $A0
17A4 338C                    FDB    $338C
17A6 42 49 54 41             FCC    "BITA"
17AA A0                      FCB    $A0
17AB 338C                    FDB    $338C
17AD 4C 44 41                FCC    "LDA"
17B0 A0 A0                   FCB    $A0,$A0
17B2 338C                    FDB    $338C
17B4 53 54 41                FCC    "STA"
17B7 A0 A0                   FCB    $A0,$A0
17B9 338C                    FDB    $338C
17BB 45 4F 52 41             FCC    "EORA"
17BF A0                      FCB    $A0
17C0 338C                    FDB    $338C
17C2 41 44 43 41             FCC    "ADCA"
17C6 A0                      FCB    $A0
17C7 338C                    FDB    $338C
17C9 4F 52 41                FCC    "ORA"
17CC A0 A0                   FCB    $A0,$A0
17CE 338C                    FDB    $338C
17D0 41 44 44 41             FCC    "ADDA"
17D4 A0                      FCB    $A0
17D5 338C                    FDB    $338C
17D7 43 4D 50 58             FCC    "CMPX"
17DB A0                      FCB    $A0
17DC 338C                    FDB    $338C
17DE 4A 53 52                FCC    "JSR"
17E1 A0 A0                   FCB    $A0,$A0
17E3 888C                    FDB    $888C
17E5 4C 44 58                FCC    "LDX"
17E8 A0 A0                   FCB    $A0,$A0
17EA 338C                    FDB    $338C
17EC 53 54 58                FCC    " TX"
17EF A0 A0                   FCB    $A0,$A0
17F1 338C                    FDB    $338C
                     *END Bx start Cx
17F3 53 55 42 42             FCC    "SUBB"
17F7 A0                      FCB    $A0
17F8 4288                    FDB    $4288
17FA 43 4D 50 42             FCC    "CMPB"
17FE A0                      FCB    $A0
17FF 4288                    FDB    $4288
1801 53 42 43 42             FCC    "SBCB"
1805 A0                      FCB    $A0
1806 4288                    FDB    $4288
1808 41 44 44 44             FCC    "ADDD"
180C A0                      FCB    $A0
180D 438C                    FDB    $438C
180F 41 4E 44 42             FCC    "ANDB"
1813 A0                      FCB    $A0
1814 4288                    FDB    $4288
1816 42 49 54 42             FCC    "BITB"
181A A0                      FCB    $A0
181B 4288                    FDB    $4288
181D 4C 44 42                FCC    "LDB"
1820 A0 A0                   FCB    $A0,$A0
1822 4288                    FDB    $4288
1824                         ILLEG
182B 45 4F 52 42             FCC    "EORB"
182F A0                      FCB    $A0
1830 4288                    FDB    $4288
1832 41 44 43 42             FCC    "ADCB"
1836 A0                      FCB    $A0
1837 4288                    FDB    $4288
1839 4F 52 42                FCC    "ORB"
183C A0 A0                   FCB    $A0,$A0
183E 4288                    FDB    $4288
1840 41 44 44 42             FCC    "ADDB"
1844 A0                      FCB    $A0
1845 4288                    FDB    $4288
1847 4C 44 44                FCC    "LDD"
184A A0 A0                   FCB    $A0,$A0
184C 438C                    FDB    $438C
184E                         ILLEG
1855 4C 44 55                FCC    "LDU"
1858 A0 A0                   FCB    $A0,$A0
185A 438C                    FDB    $438C
185C                         ILLEG
                     *END Cx START Bx
1863 53 55 42 42             FCC    "SUBB"
1867 A0                      FCB    $A0
1868 2288                    FDB    $2288
186A 43 4D 50 42             FCC    "CMPB"
186E A0                      FCB    $A0
186F 2288                    FDB    $2288
1871 53 42 43 42             FCC    "SBCB"
1875 A0                      FCB    $A0
1876 2288                    FDB    $2288
1878 41 44 44 44             FCC    "ADDD"
187C A0                      FCB    $A0
187D 2288                    FDB    $2288
187F 41 4E 44 42             FCC    "ANDB"
1883 A0                      FCB    $A0
1884 2288                    FDB    $2288
1886 42 49 54 42             FCC    "BITB"
188A A0                      FCB    $A0
188B 2288                    FDB    $2288
188D 4C 44 42                FCC    "LDB"
1890 A0 A0                   FCB    $A0,$A0
1892 2288                    FDB    $2288
1894 53 54 42                FCC    "STB"
1897 A0 A0                   FCB    $A0,$A0
```

```
1899 2288                    FDB    $2288
189B 45 4F 52 42             FCC    "EORB"
189F A0                      FCB    $A0
18A0 2288                    FDB    $2288
18A2 41 44 43 42             FCC    "ADCB"
18A6 A0                      FCB    $A0
18A7 2288                    FDB    $2288
18A9 4F 52 42                FCC    "ORB"
18AC A0 A0                   FCB    $A0,$A0
18AE 2288                    FDB    $2288
18B0 41 44 44 42             FCC    "ADDB"
18B4 A0                      FCB    $A0
18B5 2288                    FDB    $2288
18B7 4C 44 44                FCC    "LDD"
18BA A0 A0                   FCB    $A0,$A0
18BC 2288                    FDB    $2288
18BE 53 54 44                FCC    "STD"
18C1 A0 A0                   FCB    $A0,$A0
18C3 2288                    FDB    $2288
18C5 4C 44 55                FCC    "LDU"
18C8 A0 A0                   FCB    $A0,$A0
18CA 2288                    FDB    $2288
18CC 53 54 55                FCC    "STU"
18CF A0 A0                   FCB    $A0,$A0
18D1 2288                    FDB    $2288
                     *END Dx START Ex
18D3 53 55 42 42             FCC    "SUBB"
18D7 A0                      FCB    $A0
18D8 1288                    FDB    $1288
18DA 43 4D 50 42             FCC    "CMPB"
18DE A0                      FCB    $A0
18DF 1288                    FDB    $1288
18E1 53 42 43 42             FCC    "SBCB"
18E5 A0                      FCB    $A0
18E6 1288                    FDB    $1288
18E8 41 44 44 44             FCC    "ADDD"
18EC A0                      FCB    $A0
18ED 1288                    FDB    $1288
18EF 41 4E 44 42             FCC    "ANDB"
18F3 A0                      FCB    $A0
18F4 1288                    FDB    $1288
18F6 42 49 54 42             FCC    "BITB"
18FA A0                      FCB    $A0
18FB 1288                    FDB    $1288
18FD 4C 44 42                FCC    "LDB"
1900 A0 A0                   FCB    $A0,$A0
1902 1288                    FDB    $1288
1904 53 54 42                FCC    "STB"
1907 A0 A0                   FCB    $A0,$A0
1909 1288                    FDB    $1288
190B 45 4F 52 42             FCC    "EORB"
190F A0                      FCB    $A0
1910 1288                    FDB    $1288
1912 41 44 43 42             FCC    "ADCB"
1916 A0                      FCB    $A0
1917 1288                    FDB    $1288
1919 4F 52 42                FCC    "ORB"
191C A0 A0                   FCB    $A0,$A0
191E 1288                    FDB    $1288
1920 41 44 44 42             FCC    "ADDB"
1924 A0                      FCB    $A0
1925 1288                    FDB    $1288
1927 4C 44 44                FCC    "LDD"
192A A0 A0                   FCB    $A0,$A0
192C 1288                    FDB    $1288
192E 53 54 44                FCC    "STD"
1931 A0 A0                   FCB    $A0,$A0
1933 1288                    FDB    $1288
1935 4C 44 55                FCC    "LDU"
1938 A0 A0                   FCB    $A0,$A0
193A 1288                    FDB    $1288
193C 53 54 55                FCC    "STU"
193F A0 A0                   FCB    $A0,$A0
1941 1288                    FDB    $1288
                     *END Ex START Fx
1943 53 55 42 42             FCC    "SUBB"
1947 A0                      FCB    $A0
1948 338C                    FDB    $338C
194A 43 4D 50 42             FCC    "CMPB"
194E A0                      FCB    $A0
194F 338C                    FDB    $338C
1951 53 42 43 42             FCC    "SBCB"
1955 A0                      FCB    $A0
1956 338C                    FDB    $338C
1958 41 44 44 44             FCC    "ADDD"
195C A0                      FCB    $A0
195D 338C                    FDB    $338C
195F 41 4E 44 42             FCC    "ANDB"
1963 A0                      FCB    $A0
1964 338C                    FDB    $338C
1966 42 49 54 42             FCC    "BITB"
196A A0                      FCB    $A0
196B 338C                    FDB    $338C
196D 4C 44 42                FCC    "LDB"
1970 A0 A0                   FCB    $A0,$A0
1972 338C                    FDB    $338C
1974 53 54 42                FCC    " TB"
1977 A0 A0                   FCB    $A0,$A0
1979 338C                    FDB    $338C
197B 45 4F 52 42             FCC    "EORB"
197F A0                      FCB    $A0
1980 338C                    FDB    $338C
1982 41 44 43 42             FCC    "ADCB"
1986 A0                      FCB    $A0
1987 338C                    FDB    $338C
1989 4F 52 42                FCC    "ORB"
```

```
198C A0 A0                   FCB    $A0,$A0
198E 338C                    FDB    $338C
1990 41 44 44 42             FCC    "ADDB"
1994 A0                      FCB    $A0
1995 338C                    FDB    $338C
1997 4C 44 44                FCC    "LDD"
199A A0 A0                   FCB    $A0,$A0
199C 338C                    FDB    $338C
199E 53 54 44                FCC    "STD"
19A1 A0 A0                   FCB    $A0,$A0
19A3 338C                    FDB    $338C
19A5 4C 44 55                FCC    "LDU"
19A8 A0 A0                   FCB    $A0,$A0
19   338C                    FDB    $338C
19AC 53 54 55                FCC    "STU"
19AF A0 A0                   FCB    $A0,$A0
19B1 338C                    FDB    $338C
               19B3  ZZZZY   EQU    *
19B3 4C 42 52 4E     TENCMD  FCC    "LBRN"
19B7 A0                      FCB    $A0
19B8 D308                    FDB    $D308
19BA 4C 42 48 49             FCC    "LBHI"
19BE A0                      FCB    $A0
19BF D308                    FDB    $D308
19C1 4C 42 4C 53             FCC    "LBLS"
19C5 A0                      FCB    $A0
19C6 D308                    FDB    $D308
19C8 4C 42 43 43             FCC    "LBCC"
19CC A0                      FCB    $A0
19CD D308                    FDB    $D308
19CF 4C 42 43 53             FCC    "LBCS"
19D3 A0                      FCB    $A0
19D4 D308                    FDB    $D308
19D6 4C 42 4E 45             FCC    "LBNE"
19DA A0                      FCB    $A0
19DB D308                    FDB    $D308
19DD 4C 42 45 51             FCC    "LBEQ"
19E1 A0                      FCB    $A0
19E2 D308                    FDB    $D308
19E4 4C 42 56 43             FCC    "LBVC"
19E8 A0                      FCB    $A0
19E9 D308                    FDB    $D308
19EB 4C 42 56 53             FCC    "LBVS"
19EF A0                      FCB    $A0
19F0 D308                    FDB    $D308
19F2 4C 42 50 4C             FCC    "LBPL"
19F6 A0                      FCB    $A0
19F7 D308                    FDB    $D308
19F9 4C 42 4D 49             FCC    "LBMI"
19FD A0                      FCB    $A0
19FE D308                    FDB    $D308
1A00 4C 42 47 45             FCC    "LBGE"
1A04 A0                      FCB    $A0
1A05 D308                    FDB    $D308
1A07 4C 42 4C 54             FCC    "LBLT"
1A0B A0                      FCB    $A0
1A0C D308                    FDB    $D308
1A0E 4C 42 47 54             FCC    "LBGT"
1A12 A0                      FCB    $A0
1A13 D308                    FDB    $D308
1A15 4C 42 4C 45             FCC    "LBLE"
1A19 A0                      FCB    $A0
1A1A D308                    FDB    $D308
1A1C 53 57 49 32             FCC    "SWI2"
1A20 A0                      FCB    $A0
1A21 01C0                    FDB    $01C0
1A23 43 4D 50 44             FCC    "CMPD"
1A27 A0                      FCB    $A0
1A28 4308                    FDB    $4308
1A2A 43 4D 50 59             FCC    "CMPY"
1A2E A0                      FCB    $A0
1A2F 4308                    FDB    $4308
1A31 4C 44 59                FCC    "LDY"
1A34 A0 A0                   FCB    $A0,$A0
1A36 4308                    FDB    $4308
1A38 43 4D 50 44             FCC    "CMPD"
1A3C A0                      FCB    $A0
1A3D 2200                    FDB    $2200
1A3F 43 4D 50 59             FCC    "CMPY"
1A43 A0                      FCB    $A0
1A44 2200                    FDB    $2200
1A46 4C 44 59                FCC    "LDY"
1A49 A0 A0                   FCB    $A0,$A0
1A4B 2200                    FDB    $2200
1A4D 53 54 59                FCC    "STY"
1A50 A0 A0                   FCB    $A0,$A0
1A52 2200                    FDB    $2200
1A54 43 4D 50 44             FCC    "CMPD"
1A58 A0                      FCB    $A0
1A59 1200                    FDB    $1200
1A5B 43 4D 50 59             FCC    "CMPY"
1A5F A0                      FCB    $A0
1A60 1200                    FDB    $1200
1A62 4C 44 59                FCC    "LDY"
1A65 A0 A0                   FCB    $A0,$A0
1A67 1200                    FDB    $1200
1A69 53 54 59                FCC    "STY"
1A6C A0 A0                   FCB    $A0,$A0
1A6E 1200                    FDB    $1200
1A70 43 4D 50 44             FCC    "CMPD"
1A74 A0                      FCB    $A0
1A75 3308                    FDB    $3308
1A77 43 4D 50 59             FCC    "CMPY"
1A7B A0                      FCB    $A0
1A7C 3308                    FDB    $3308
1A7E 4C 44 59                FCC    "LDY"
```

```
1A81 A0 A0                    FCB    $A0,$A0
1A83 3308                     FDB    $3308
1A85 53 54 59                 FCC    "STY"
1A88 A0 A0                    FCB    $A0,$A0
1A8A 3308                     FDB    $3308
1A8C 4C 44 53                 FCC    "LDS"
1A8F A0 A0                    FCB    $A0,$A0
1A91 4308                     FDB    $4308
1A93 4C 44 53                 FCC    "LDS"
1A96 A0 A0                    FCB    $A0,$A0
1A98 2200                     FDB    $2200
1A9A 53 54 53                 FCC    "STS"
1A9D A0 A0                    FCB    $A0,$A0
1A9F 2200                     FDB    $2200
1AA1 4C 44 53                 FCC    "LDS"
1AA4 A0 A0                    FCB    $A0,$A0
1AA6 1200                     FDB    $1200
1AA8 53 54 53                 FCC    "STS"
1AAB A0 A0                    FCB    $A0,$A0
1AAD 1200                     FDB    $1200
1AAF 4C 44 53                 FCC    "LDS"
1AB2 A0 A0                    FCB    $A0,$A0
1AB4 3308                     FDB    $3308
1AB6 53 54 53                 FCC    "STS"
1AB9 A0 A0                    FCB    $A0,$A0
1ABB 3308                     FDB    $3308
1ABD 53 57 49 33     ELVCMD   FCC    "SWI3"
1AC1 A0                       FCB    $A0
1AC2 01C0                     FDB    $01C0
1AC4 43 4D 50 55              FCC    "CMPU"
1AC8 A0                       FCB    $A0
1AC9 4308                     FDB    $4308
1ACB 43 4D 50 53              FCC    "CMPS"
1ACF A0                       FCB    $A0
1AD0 4308                     FDB    $4308
1AD2 43 4D 50 55              FCC    "CMPU"
1AD6 A0                       FCB    $A0
1AD7 2200                     FDB    $2200
1AD9 43 4D 50 53              FCC    "CMPS"
1ADD A0                       FCB    $A0
1ADE 2200                     FDB    $2200
1AE0 43 4D 50 55              FCC    "CMPU"
1AE4 A0                       FCB    $A0
1AE5 1200                     FDB    $1200
1AE7 43 4D 50 53              FCC    "CMPS"
1AEB A0                       FCB    $A0
1AEC 1200                     FDB    $1200
1AEE 43 4D 50 55              FCC    "CMPU"
1AF2 A0                       FCB    $A0
1AF3 3308                     FDB    $3308
1AF5 43 4D 50 53              FCC    "CMPS"
1AF9 A0                       FCB    $A0
1AFA 3308                     FDB    $3308
                              OPT    LIS

                     *DUMPS TAG TABLE FOR ANALYSIS.
                     *ACCESSED BY A *D AFTER THE
                     *REST OF THE COMMAND LINE.
                     *NOT DOCUMENTED.  IF YOU FOUND
                     *IT YOU CAN USE IT, BUT IT WAS
                     *MEANT FOR DEBUGGING THE PROGRAM.
                     *
1AFC 9E    07        DUMP     LDX    TAGST      *DEBUG
1AFE C6    07        DUMP1    LDB    #7
1B00 9C    09        DUMP2    CMPX   TAGEND
1B02 27    30                 BEQ    DUMP4
1B04 BD    CD3C               JSR    OUTHEX
1B07 A6    80                 LDA    0,X+
1B09 26    02                 BNE    DUMP3
1B0B 86    20                 LDA    #$20
1B0D BD    CD18      DUMP3    JSR    PUTCHR
1B10 86    20                 LDA    #$20
1B12 BD    CD18               JSR    PUTCHR
1B15 34    10                 PSHS   X
1B17 BD    CD45               JSR    OUTADR
1B1A 35    10                 PULS   X
1B1C 30    02                 LEAX   2,X
1B1E 86    20                 LDA    #$20
1B20 BD    CD18               JSR    PUTCHR
1B23 86    20                 LDA    #$20
1B25 BD    CD18               JSR    PUTCHR
1B28 5A                       DECB
1B29 26    D5                 BNE    DUMP2
1B2B 34    10                 PSHS   X
1B2D BD    CD24               JSR    PCRLF
1B30 35    10                 PULS   X
1B32 20    CA                 BRA    DUMP1
1B34 BD    CD24      DUMP4    JSR    PCRLF
1B37 39                       RTS

                     * PRINT "FCC" LINE
1B38 0D    15        FCCPRT   TST    FCCCNT
1B3A 27    1E                 BEQ    FCCPR3     *NOTHING TO PRINT
1B3C 86    22                 LDA    #$22       *CLOSF QUOTE

1B3E A7    9F 0016            STA    [FCCLOC]
1B42 8E    009A               LDX    #OLINE
1B45 108E  0060               LDY    #PADDR
1B49 C6    38                 LDB    #56
1B4B A6    80        FCCPR1   LDA    0,X+
1B4D A7    A0                 STA    0,Y+
1B4F 5A                       DECB
1B50 26    F9                 BNE    FCCPR1
1B52 17    F4BD               LBSR   PRINT
1B55 26    07                 BNE    FCCPR2
1B57 17    F4AB               LBSR   CLRPLN
```

```
1B5A 8D    03        FCCPR3   BSR    CLRDLN
1B5C 85    00                 BITA   #0
1B5E 39              FCCPR2   RTS
                     *
1B5F 86    A0        CLRDLN   LDA    #$A0
1B61 C6    38                 LDB    #56
1B63 8E    009A               LDX    #OLINE
1B66 A7    80        CLRFC1   STA    0,X+
1B68 5A                       DECB
1B69 26    FB                 BNE    CLRFC1
1B6B D7    15                 STB    FCCCNT
1B6D CC    00BA               LDD    #OOPND+1
1B70 DD    16                 STD    FCCLOC
1B72 39                       RTS
                     *

                     *MOVES FILE NAME FROM 0,X TO 0,Y.
1B73 A6    80        MOVENM   LDA    0,X+
1B75 27    02                 BEQ    MOVEN2
1B77 A7    A0                 STA    0,Y+
1B79 5A              MOVEN2   DECB
1B7A 26    F7                 BNE    MOVENM
1B7C C6    2E                 LDB    #'.
1B7E E7    A0                 STB    0,Y+
1B80 C6    03                 LDB    #3
1B82 A6    80        MOVEN1   LDA    0,X+
1B84 A7    A0                 STA    0,Y+
1B86 5A                       DECB
1B87 26    F9                 BNE    MOVEN1
1B89 39                       RTS
                     *
                     *
                     *
1B8A 9E    07        TAGCLN   LDX    TAGST      *CONDENSES TAG TABLE
1B8C 1F    12                 TFR    X,Y
1B8E 109C  09        TAGCL1   CMPY   TAGEND
1B91 27    12                 BEQ    TAGCL3     *DONE
1B93 A6    A0                 LDA    0,Y+
1B95 85    21                 BITA   #$21       *REFERENCED OR 1ST
1B97 27    08                 BEQ    TAGCL2     *NOT NEEDED
1B99 84    FE                 ANDA   #$FE
1B9B A7    80                 STA    0,X+
1B9D EC    A4                 LDD    0,Y
1B9F ED    81                 STD    0,X++
1BA1 31    22        TAGCL2   LEAY   2,Y
1BA3 20    E9                 BRA    TAGCL1
1BA5 9F    09        TAGCL3   STX    TAGEND
1BA7 9F    0B                 STX    TTGST
1BA9 9F    0D                 STX    TTGEND
1BAB 30    06                 LEAX   6,X
1BAD 9F    42                 STX    XRST
1BAF 9F    44                 STX    XREND
1BB1 39                       RTS

                     * GET KNOWN DATA AREAS
                     *
1BB2 B6    CC22      GETDAT   LDA    OUTSW      *SAVE TTY PARMS
1BB5 F6    CC1A               LDB    CURLN
1BB8 34    06                 PSHS   D
1BBA 7C    CC22               INC    OUTSW      *GO TO TERM
1BBD 8E    1C16               LDX    #MSG5
1BC0 BD    C01E               JSR    PSTRNG
1BC3 9E    03                 LDX    DATST
1BC5 9F    05                 STX    DATEND
1BC7 86    3F        GETD1    LDA    #'?
1BC9 BD    CD18               JSR    PUTCHR
1BCC BD    CD1B               JSR    INBUFF     *GET DATA
1BCF BD    CD42               JSR    GETHEX     *GET 1ST ADDR (FROM)
1BD2 25    2B                 BCS    GETD2      *NOT HEX: EXIT
1BD4 109E  05                 LDY    DATEND
1BD7 AF    A4                 STX    0,Y
1BD9 BD    CD42               JSR    GETHEX     *2ND ADDR (TO)
1BDC 25    21                 BCS    GETD2
1BDE 109E  05                 LDY    DATEND
1BE1 31    24                 LEAY   4,Y
1BE3 AF    3E                 STX    -2,Y
1BE5 EC    3E                 LDD    -2,Y       *CHECK SEQUENCE OF INPUT
1BE7 10A3  3C                 CMPD   -4,Y
1BEA 25    08                 BCS    GETD3      *NOT RIGHT!
1BEC 109F  05                 STY    DATEND
1BEF BD    CD24               JSR    PCRLF
1BF2 20    D3                 BRA    GETD1
1BF4 BD    CD24      GETD3    JSR    PCRLF
1BF7 8E    1C56               LDX    #MSG6
1BFA BD    CD1E               JSR    PSTRNG
1BFD 20    C8                 BRA    GETD1
1BFF 9E    05        GETD2    LDX    DATEND     *ADJUST OTHER POINTERS
1C01 9F    07                 STX    TAGST
1C03 30    03                 LEAX   3,X
1C05 9F    09                 STX    TAGEND
1C07 30    03                 LEAX   3,X
1C09 9F    0B                 STX    TTGST
1C0B 9F    0D                 STX    TTGEND
1C0D 35    06                 PULS   D
1C0F B7    CC22               STA    OUTSW
1C12 F7    CC1A               STB    CURLN
1C15 39                       RTS
                     *
1C16 45 4E 54 45     MSG5     FCC    "ENTER KNOWN DATA AREAS "
1C1A 52 20 4B 4E
1C1E 4F 57 4E 20
1C22 44 41 54 41
1C26 20 41 52 45
1C2A 41 53 20
1C2D 28 4F 52 20              FCC    "(OR 'END')"
1C31 27 45 4E 44
```

For the month of Aug. only!
OS-9 w/Edit.Asm.Debugger for ELEKTRA $175.00
OS-9 L1.2 for SSB $150.00

FOR THE ELEKTRA

- OS-9™ with Editor, Assembler, and Debugger **$250.00**
- STAR-DOS'™ Level 1 (FLEX™ compatible but with up to 10 active drives; (i.e. 4 8" floppies, 4 5" floppies, and 2 Winchester drives) **$75.00**
- STAR-DOS'™ Adaptation Guide **$50.00**

Read/Write Radio Shack OS-9 now available for your ELEKTRA

FOR OS-9™ by Epstein Associates

- Super Modem Program with autodial, configuration file, etc. (Available exclusively through AAA Chicago Computer Center) **$100.00**

8" Floppy Drive Special w/manual , 90 day warranty
Siemens FDD 100-8 (SSDD) $135.00
Siemens FDD 200-8 (DSDD) $185.00

THE ELEKTRA SUPER FLOPPY CONTROLLER

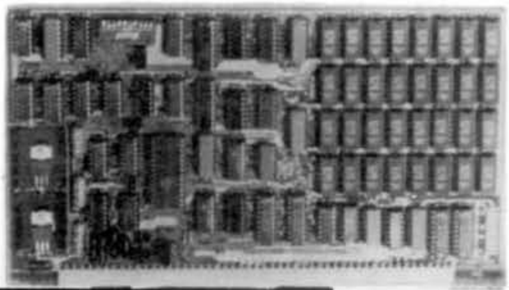
- Emulates the DC-1, DC-2, DC-3, DC-4 as well as the GIMIX #28, #38, #48, and #58 controllers **$295.00**

Removable cartridge Winchester now available for your ELEKTRA

RACKMOUNT ELEKTRA COMPUTER CABINET

- 17" W x 21.5" D x 6.7" H
  Holds two half height 5¼" floppy drives **$250.00**

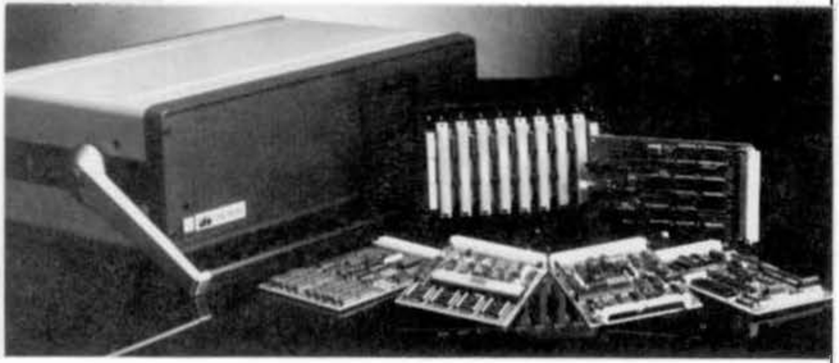
1200 Baud Auto Answer Modem $199.00



2MHz 256K Memory Board
with on board DAT by
COMPUTER EXCELLENCE, INC.
$749.00

68000 - VME Bus

Mizar Development System
Includes:
256K RAM
1 360K 5" Floppy
1 10 MByte Winchester
4 Serial Ports
Synchronous and/or
Asynchronous
Screen Editor
Assembler
"C" Compiler
SASI Interface
$6495.00 plus shipping



Phone:
AAA Chicago Computer Center
Technical Consultation available most weekdays from 4 p.m. to 6 p.m. CST
(312) 459-0450 120 Chestnut Lane Wheeling, IL 60090
See our catalog and ordering information on the next page.

ELEKTRA COMPUTER SYSTEM Includes chassis, dual port serial interface with two cables, CPU 8/9, 4K Humbug, 56K static RAM, super floppy controller with inboard ribbon cable, Star-Dos, dual 80 track DSDD floppy drives (other combinations available; phone) $2795.00

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! M de of heavyweight 0.090" thick aluminum. Interi r is 18-1/2" wide by 21-7/8" deep by 6-3/4" high Heavy du ty A.C. line cord A C fuse holder EMI filter Fan with filter Back panel has 10 cutouts for 'D' type data connectors Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives $250 00

RACKMOUNT E KTRA COMPUTER CABINET 17"w x 21.5"d x 6.7"h Holds two half height 5-1/4" floppy drives $250 00

Filler Plate for 5-1/4" drive opening $10.00 Fan Filter $10 00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v Multi-tapped primary for fine tuning $200 00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive $75 00

AUXILIARY POWER SUPPLY to power second Winchester drive $125 00

ELEKTRA UNIVERSAL SS-50/SS-50C MOTHERBOARD Heavyweight 0 125" thick 18" long by 9" wide 11 memory (50 pin) slots 8 I/O (30 pin) slots Complete address decoding and selection, as well as extended address capability for I/O slots Choice of 4, 8, or 16 addresses per I/O slot 1" spacing between all memory and I/O slots On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines slow device circuitry permitting 1 Mhz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.

Mounting hardware	$5.00	Bareboard w/documentation	$80 00
Kit w/gold connectors	$400 00	Assembled w/gold connectors	$460 00
Kit w/tin connectors	$320 00	Asembled w/tin connectors	$380 00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold uare pin connectors, assembled and teste $950 00

ELEKTRA CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809 Has provision for up to 3 2716 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges Run OS-9™, FLEX™, STAR-DOS
Bareboard $50 00 Kit $225 00 Assembled $275 00
Optional Baud Rate Generator $25 00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot Can be configured for 4 or 16 addresses per port RTS, CTS, DTR, DCD, IRQ, FIRQ/NMI, and baud rate can be appropriately implemented for each port
Bareboard $25.00 Kit $75.00 Assembled $95 00
Cable with jack socket assembies (two needed per board) Each $25 00

E KTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot Can be configured for 4 or 16 addre ses per I/O slot The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line
Bareboard $25 00 Kit $60 00 Assembled $80 00
Cable with jack socket assemblies (two needed per board) Each $25 00

ELEKTRA 64K STA IC RAM/ROM MEMORY BOARDS with gold connectors (tin available) Assembled and tested With 56K RAM $289 0 With 64K RAM $299 00
Kit With 56K RAM $219 00 With 64K RAM $249 00

ELE TRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives (FLEX system limit is four drives) Single density or double density, 1MHz or 2MHz, 6800 or 6809 (Double density 8" must be at 2MHz, all other combinations of performance are possible.) Analog phase locked loop data separators with separate adjustments for 5" and 8" drives Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC Assembled and tested $295 00
Disk with drivers, setup, and formatting utilities Specify FLEX 2 0, 6800 Gen FLEX, FLEX 9.0, FLEX 9 1, or 6809 Gen FLEX, 5" or 8" $30 00
Disk with drivers for OS-9 (Specify 5" or 8") $50 00
STAR-DOS (Specify ELEKTRA or DC-2, 5" or 8") $75.00 Adaptation guide $50.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors SS-50 bus, extended addressing capabilities, DMA on board sector buffer, drivers included for 6809 FLEX or OS-9 Specify whose version of FLEX that you are using Drivers for FLEX2 (6800) are available for an additional $100 00 Price includes host interface, controller, drive(s), and cables

7 Megabyte single drive sys.	$1995 00	14 Megabyte dual drive sys	$2995 00
12 Megabyte single drive sys.	$2295 00	24 Megabyte dual drive sys	$3595 00
19 Megabyte single drive sys	$2995 00	38 Megabyte dual drive sys	$4695 00

(19 Megabyte drives are the largest that can be supported by FLEX)
Circuit boards, cables, software (No drives) 995 00
SS-50C DMA Bus Interface board only 695 00

ELEKTRA HD-5 Cabinet for dual 5 1/4" floppy drives with power supply, line co d, fuse power switch, and power cables to drives 150.00

ELEKTRA HD-5W As above but with EMI filter, fan, and heavy duty power supply Powers 1 floppy and 1 Winchester 199 00

5' ribbon cable for dual outboard 5-1/4" disk drives 40 00
2' ribbon cable for dual inboard 5-1/4" disk drives 35 00
Custom cables available Phone

ELEKTRA HD-8 Dual 8" drive cabinet, EMI filter, fan with filter, power supply and power supply cables 350 00

6' ribbon cable for dual 8" disk drives 45 00

ELEK RA 30 PIN PROTOTYPING BOARD 20 00
ELEKTRA 50 PIN PROTOTYPING BOARD 40 00

GOLD 10 PIN CONNECTORS (Specify male with square pins or female) 1 50
TIN 10 PIN CONNECTORS (Specify male with square pins or female) 50

ELEKTRA is a trademark of AAA Chicago Computer Center
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc
HELIX is a trademark of Hazelwood Computer Systems
OS-9 and **BASIC09** are trademarks of Motorola Inc and Microware Systems Corp

AAA CHICAGO COMPUTER CENTER (312) 459-0450
120 CHESTNUT LANE • WHEELING, IL 60090
Technical consultation vailable 4 PM to 6 PM most weekdays Closed evenings and weekends

TERMS Minimum order $20.00 Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50) Illinois residents add 7% sales tax We will refund your overestimated shipping and handling charges Foreign shipping and handling add 10% (MINIMUM $10 00) Foreign orders must be prepaid in U S dollars Checks must be drawn on a U S bank Heavy foreign items will be shipped air freight collect Please phone between 4 PM and 6 PM weekdays if questions arise regarding shippi g fees Master Charge, Visa, and American Express honored

Our apology We are not staffed to answer technical inquiries through the mail Please phone for technical help during the hours indicated above The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog Our ads are intended to serve that purpose Prices and inventory are subject to change without advance notice

SUPER MODEM PROGRAM Single character commands No interrupts required Transmit manually or transmit disk files (text) of any length to distant computer Receive and save disk files (text) on local disk system X-on X-off supported Tested for full duplex at speeds up to 9600 baud Half duplex optio cl o option Replaces CR with CR/LF (user option) Slow disk file transmit option

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8"
Instruction Manual and disk with both source and object code $75 00

OS-9 Super Modem Program by Epstein Associates with autodial, configuration file, etc 100 00

ALL IN ONE
Editor — Text Pr ossor — Mailing Labels — Mailing Lists — Multiple Form Letters
Use any CRT terminal and printer — Best Package For The Money Anywhere!

Specify 6800 or 6809, SSB or FLEX™, 5" or 8" 75 00
Add $35 00 for printed source listing; add $100 for source on disk
All-In-One, Write'n spell, and Spell'n Fix package 250 00

Software by Technical Systems Consultants, Inc.	FLEX™ Source (List)	FLEX™ Source (Disk)	FLEX™ Man. Only	FLEX™ Object w/Man.	UniFLEX™ Add. Man.	UniFLEX™ Man. Only	UniFLEX™ Object w/Man
Gen FLEX w/Edit & ASMB	—	—	—	250	—	—	—
FLEX 9 1 (DC-2) w/Edit & ASMB	—	—	25	150	40	100	550
Advanced Programmers Guide	—	—	25	—	—	—	—
Editor	100	250	25	50	—	—	—
Assembler	150	250	25	50	—	—	—
Debug	175	250	25	75	—	—	—
Extended Basic	—	—	25	100	20	50	200
Basic Precompiler	—	—	25	50	10	25	150
Sort/Merge	—	—	25	75	20	35	150
Utilities	—	Inc	25	75	10	25	150
Diagnostics	—	—	25	75	—	—	—
Text Proce sor	150	250	25	75	20	35	150
68000 X-ASMB on 6809	—	—	25	250	20	35	300
Pascal	—	—	50	200	25	50	300
Rel ASMB/Linking Loader	—	—	25	150	20	35	175
6800 X-ASMB on 6809	—	—	—	100	—	—	—
Cobol	—	—	—	—	30	75	750
Fortran 77	—	—	—	—	35	65	450

Software by Microware Systems Corp. (Suggested List Prices, varies/mdg)	Run-Time Package	Source	Manual Only	Object w/Man.
OS-9™ Level 1 w/Edit, Asm Debug		400 00	40 00	250 00
OS-9™ Level 2 w/Edit, Asm, Debug		400 00	40 00	500 00
OS-9™ Edit, Asm, Debug Pkg			25 00	125 00
Device Driver for Disk Controller (Specify Model)		100 00		
Device Driver for ACIA and PIA		50 00		
Clock Driver for 6840 and 58167 clock chips		35 00		
Entertainment Pack I, or File Handler Toolbox or NinoCom			10 00	85 00
Print Spooler (Level 2 only)			15 00	95 00
Virtual Disk Driver (Level 2 only)			10 00	85 00
RMA Relocatable Macro Assembler			20 00	125 00
RMA/68000 Cross Assembler				400 00
BASIC09™ W/Run-Time	50 00	N/A	25 00	200 00
BASIC09 Tour Guide Book			18 95	
"C" Compiler			25 00	250 00
C Programming Language (Kernighan & Ritchie)			19 95	
CIS Cobol Compiler w/Forms 2 Prog Gen	50 00	N/A	40 00	400 00
Pascal Compiler	50 00	N/A	25 00	250 00
Sage Application Generator	300 00	N/A	25 00	995 00
Microware yearly support service (All products)				150 00

Edition Update wo/manuals 25 00 Version Update w/manuals 75 00

Special Software
STAR-DOS Level 1 (Specify ELEKTRA or DC-2) $75 00 Adaption guide $50 00
2K MICROBUG 40 00 4K HUMBUG 75 00 Custom versions $85 00
Spell'n Fix by Peter Stark 178 58 Write'n Spell by Peter Stark 75 11
All-In-One, Spell'n Fix, and Write'n Spell package 250 00
SUPER SLEUTH Dis ssembler System ($101 00 for OS-9 version) 98 00

SD/DD DISK DRIVES 30 day guarantee	1 head Tandon	2 heads Tandon	2 heads CDC	1 head MPI	2 heads MPI
5-1/4", 40 tracks	225 00	300 00	300 00	250 00	325.00
5-1/4", 80 tracks	300 00	375 00	375 00	325 00	400 00

MPI or CDC Service Manual (S cify 40 or 80 track) 26 00 Qume DT-8 550 00

OUTBOARD EPROM PROGRAMMERS BY OPTIMAL TECHNOLOGY
Model EP-2A-79 (Personality modules extra) 169 00
Optimal Technology, Inc 30 pin parallel I/O board for EP-2A-79 37 00
FLEX™ Software package for EP-2A-79 (Specify 6800 or 6809) 30 00
OS-9 Software package for EP-2A-79 10 00
Model EP-2B-87 (RS-232/20 MA, Motorola fmt, 8K buffer, 1200/9600 baud) 575 00
Model EP-28-88-4 (Copies 1 to 4 EPROMS) 550 00
Personality/Copy Modules for 2708, 2716, 27C16, 2732, 27C32, 2732A, 2758, MCM68764, MCM68766, 2764, 27C64, 2764A, 27128, 27128A, 27256, 27C256, 2508, 2516, 2532, 2564, 25128, 2816, 2816A, R87C32, 8751, 38E70 $17 to $39

SPECIALS
- U.S Robotics 1200 baud direct connect auto answer modem 289 00
- SSB BFD Floppy Disk Controller (Version 3) Run FLEX or SSB DOS 100 00
- SWTPC 4K Memory $15 00 MP-Mb (4K bareboard) 9 95
- SWTPC DMF2 $598.00 S-32' RAM not included 124 50
- SWTPC MP-09 2MHz CPU' $295 00 'While supplies last
- High speed tape reader 50 00 300 Baud acoustic modem 129 00
- TI 810 Printer w/lower case and full vertical forms control 1200 00

SPECIAL BOARDS
Microtime II Calendar and Clock Board (Assembled) 60 00
Data Mart 16K EPROM bareboard (2708 chips) 30 00

Smoke Signal Broadcasting
DCB-4A Double Density Controller Board for 5" and 8" with DOS 549 00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8") 76 00
SSB version of FLEX™ (While supply lasts) 150 00
LMB-1A Motherboard 389 00
SCB-69 6809 CPU Board 399 00
Chief 9524 64K Computer System with OS-DT-DD 5" FD 4325 00
Dynamic Memory Bo rds M-128-X 995 00 M-256-X 1295 00 M-512-X 1885 00

HELIX
64K 6809 Computer $2395 00 Other computer systems available
DMA 5" and 8" Floppy Controller 495 00 6809 CPU Board 495 00
68008 board for SS-50 595 00 CP/M-68K 350 00

GIMIX CLEARANCE SALE	LIST	OUR PRICE		LIST	OUR PRICE
C ble (Par I/O)	24.95	20 00	6800 CPU board	224 03	100.00
64 X 16 Video Boards	198 71	100.00	Single prtser, 1 cable	113 36	90 00
16K Mem Bds w/cntrl reg		145 00	Dual prt par, 2 cables	138 32	110 00
93L422 DAT chip	17 50	15 00	**80x24 Video Boards**	**398.76**	**250.00**

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits AAA Chicago Computer Center does warranty and maintain service for our assembled boards The customer should carefully take into consideration the small differential separating our kit and a embled prices when making his choice of purchase

```
1C35 27 29
1C37 00 0A 00 00          FCB    $D,$A,0,0,0,0
1C3B 00 00
1C3D 46 52 4F 4D          FCC    "FROM,TO   (INCLUSIVE)"
1C41 2C 54 4F 20
1C45 20 20 28 49
1C49 4E 43 4C 55
1C4D 53 49 56 45
1C51 29
1C52 000A 0A04            FDB    $00A,$A04
1C56 07            MSG6   FCB    7            *RING BELL
1C57 4E 4F 21 20          FCC    "NO!  1ST ADDR > 2ND ADDR!"
1C5B 20 31 53 54

1C5F 20 41 44 44
1C63 52 20 3E 20
1C67 32 4E 44 20
1C6B 41 44 44 52
1C6F 21
1C70 00 0A 00 00          FCB    $0,$A,0,0,0,0,4
1C74 00 00 04
              1C77 PGMEND EQU    *            *TAG TABLE OVERLAYS

                   *DOES PRELIMINARY SETUP OF FILES,
                   *ETC.  IT IS OVERLAID BY THE TAG
                   *TABLE ONCE IT HAS DONE ITS THING.
                   *
1C77 8E    0119    PRELIM LDX    #FCBIN       *POINT TO CONTROL BLOCK
1C7A BD    CD20           JSR    GETFIL       *GET FILE SPEC
1C7D 1025 00CE            LBCS   PRE3         *ERROR DETECTED
1C81 4F                   CLRA                *.BIN DEFAULT
1C82 BD    CD33           JSR    SETEXT       *SET EXTENSION
1C85 86    01             LDA    #1
1C87 A7    84             STA    FUNC,X       *OPEN FOR READ
1C89 BD    D406           JSR    FMS          *CALL FILE MGMT SYSTEM
1C8C 1026 008F            LBNE   PRE3         *ERROR
1C90 BD    D406           JSR    FMS          *ONE READ KEEPS
1C93 1026 0088            LBNE   PRE3         * FLEX HAPPY
1C97 EC    88 11          LDD    STRSEC,X
1C9A ED    88 1E          STD    NEXT,X       *ALLOW INITIAL REWIND
1C9D 8E    0259           LDX    #FCBOUT      *POINT TO OUTPUT BLOCK
1CA0 BD    CD2D           JSR    GETFIL
1CA3 24    12             BCC    PRE1         *GOOD NAME; ELSE
1CA5 108E 011C            LDY    #FCBIN+DRV *TRANSFER NAME
1CA9 30    03             LEAX   DRV,X
1CAB C6    09             LDB    #9
1CAD A6    A0      PRE2   LDA    0,Y+
1CAF A7    80             STA    0,X+
1CB1 5A                   DECB
1CB2 26    F9             BNE    PRE2
1CB4 8E    0259           LDX    #FCBOUT      *RESTORE X
1CB7 86    44      PRE1   LDA    #'D          *DSA EXTENSION
1CB9 A7    0C             STA    DRV+9,X
1CBB 86    53             LDA    #'S
1CBD A7    0D             STA    DRV+10,X
1CBF 86    41             LDA    #'A
1CC1 A7    0E             STA    DRV+11,X
1CC3 86    02      PRE10  LDA    #2
1CC5 A7    84             STA    FUNC,X       *OPEN FOR WRITE
1CC7 BD    D406           JSR    FMS
1CCA 27    1D             BEQ    PRE9
1CCC A6    01             LDA    STAT,X
1CCE 81    03             CMPA   #3
1CD0 26    70             BNE    PRE3         *ERROR
1CD2 A6    04             LDA    DRV+1,X
1CD4 A7    9F 0009        STA    [TAGEND]
1CD8 86    0C             LDA    #12          *DELETE DUPLICATE
1CDA A7    84             STA    FUNC,X
1CDC BD    D406           JSR    FMS
1CDF 26    6E             BNE    PRE3         *ERROR
1CE1 A6    9F 0009        LDA    [TAGEND]
1CE5 A7    04             STA    DRV+1,X      *RESTORE NAME
1CE7 20    DA             BRA    PRE10
1CE9 6F    88 3B   PRE9   CLR    COMP,X       *COMPRESS SPACES
1CEC 8E    011D           LDX    #FCBIN+DRV+1
1CEF 108E 00E2            LDY    #HFILE
1CF3 C6    08             LDB    #8
1CF5 17    FE7B           LBSR   MOVENM       *MOVE NAME TO HEADING
1CF8 8E    025D           LDX    #FCBOUT+DRV+1
1CFB 108E 00FC            LDY    #HOFILE
1CFF C6    08             LDB    #8
1D01 17    FE6F           LBSR   MOVENM
1D04 17    F343           LBSR   PHEAD
1D07 BE    CC14           LDX    LNBFPT       *BACK POINTER UP
1D0A 30    1B             LEAX   -5,X
1D0C BF    CC14           STX    LNBFPT
1D0F BD    CD27    PRE4   JSR    NXTCH        *GET NEXT CHAR
1D12 81    0D             CMPA   #$D
1D14 27    23             BEQ    PRE5         *C/R
1D16 B1    CC02           CMPA   TTYEOL
1D19 27    1E             BEQ    PRE5         *END OF LINE
1D1B 81    2B             CMPA   #'+
1D1D 26    F0             BNE    PRE4
1D1F BD    CD27    PRE6   JSR    NXTCH        *CHECK FOR DEBUG
1D22 81    0D             CMPA   #$D
1D24 27    13             BEQ    PRE5         *CR
1D26 B1    CC02           CMPA   TTYEOL
1D29 27    0E             BEQ    PRE5
1D2B 81    44             CMPA   #'D
1D2D 26    02             BNE    PRE7         *DEBUG
1D2F 97    1B             STA    DEBUG
1D31 81    43      PRE7   CMPA   #'C
1D33 26    EA             BNE    PRE6
1D35 97    1D             STA    RPFLAG
1D37 20    E6             BRA    PRE6
1D39 B6    CC03    PRE5   LDA    DEPTH
1D3C 97    49             STA    TTYDEP
```

```
1D3E FC    CD0F           LDD    OUTCH
1D41 1083 CD12            CMPD   OUTCH2
1D45 27    05             BEQ    PRE8         *CRT OUTPUT
1D47 96    0D             LDA    PDEPTH
1D49 B7    CC03           STA    DEPTH        *66 LINES
1D4C 1C    FE      PRE8   CLC
1D4E 39                   RTS
1 4F BD    CD3F    PRE3   JSR    RPTERR
1D52 1A    01             SEC
1D54 39                   RTS
1D55 57 49 4C 4C          FCC    "WILLIAM STOCK"
1D59 49 41 4D 20
1D5D 53 54 4F 43
1D61 4B
1D62 31 31 32 35          FCC    "1125 LOIS DRIVE"
1D66 20 4C 4F 49
1D6A 53 20 44 52
1D6E 49 56 45
1D71 43 49 4E 43          FCC    "CINCINNATI, OHIO 45237"
1D75 49 4E 4E 41
1D79 54 49 2C 20
1D7D 4F 48 49 4F
1D81 20 34 35 32
1D85 33 37
1D87 35 31 33 2D          FCC    "513-641-0181"
1D8B 36 34 31 2D
1D8F 30 31 38 31
                          END    START

1 ERROR(S) DETECTED

  SYMBOL TABLE:

  ABORTF 001C   BADOP  0023   BINB   001E   BYTCNT 002F   CLOSE  0406
  CLRDLN 1B5F   CLRFC1 1B66   CLRPL1 100C   CLRPLN 1005   CMDCNT 002E
  CMDTBL 12B3   CNTR   0011   CODFLG 0021   COMP   003B   CONT1  03DA
  CONT2  03F4   CONT3  0403   CONT4  0303   CURLN  CC1A   CURTAG 0039
  DADDR  009A   DATA   0040   DATEND 0005   DATST  0003   DCD1   09AB
  DCD2   09CB   DCD2A  0903   DCD2B  09DA   DCD3   09DF   DCD3A  0A12
  DCD3B  0A28   DCD3C  0A10   DCD3D  09E7   DCD3RT 0A4C   DCDCMD 0998
  DEBUG  001B   DEPTH  CC03   DHEX   009F   DIDX   0022   DIR    0BF3
  DIR1   0C08   DLINE  009A   DO10   0A4D   DO10A  0A79   DO10C  0A62
  DO11   0A7C   DO11A  0AA8   DO11C  0A91   DOORG  0FB2   DOORG1 0FB8
  DOPCD  0083   DOPND  0089   DPFLAG 0012   DPMSG  121C   DPMSG1 1225
  DPMSG2 1230   DPMSG3 1256   DPWARN 0E47   DRV    0003   DTAG   00AD
  DUMP   1AFC   DUMP1  1AFE   DUMP2  1B00   DUMP3  1800   DUMP4  1B34
  ELVCMD 1AB0   ENDS   1271   EQUS   1267   EXT    0C10   FCBIN  0119
  FCBOUT 0259   FCBS   126A   FCCCNT 0015   FCCLOC 0016   FCCPR1 1B4B
  FCCPR2 1B5E   FCCPR3 1B5A   FCCPRT 1B38   FCCS   1274   FIND   003E
  FIND1  0058   FIND2  0069   FIND3  007A   FINDT1 0F88   FINDT2 0F9A
  FINDT3 0FA5   FINDT4 0FA9   FINDT5 0FAB   FINDTG 0F80   FIN1   0434
  FIN1A  0425   FLEX   C000   FMS    0406   FMSCLS 0403   FOOT1  108A
  FUNC   0000   GOADDR 0035   GOCNTR 003A   GORIDX 0039   GOTRK  0037
  GET1   0E2E   GET2   0E24   GETCHR CD15   GETD1  1BC7   GETD2  1BFF
  GETD3  1BF4   GETDAT 1B82   GETFIL CD20   GETHEX CD42   GETRET 0E46
  HEAD   0003   HEADM  0099   HEXAS1 0E63   HEXAS2 0E67   HEXAS3 0E71
  HEXASC 0E5B   HEXFMT 0031   HFILE  00E2   HOFILE 00FC   HPAGE  0112
  HPGNO  0117   IDX    0AAB   IDX1   0AC1   IDX10  0B4D   IDX11  0B6D
  IDX12  0B6F   IDX13  0872   IDX14  0874   IDX15  086A   IDX16  0BAA
  IDX17  0887   IDX18  08C7   IDX19  0877   IDX2   0ACC   IDX20  0881
  IDX20A 0888   IDX20B 0897   IDX3   0ADE   IDX4   0AE8   IDX5   0AFB
  IDX6   0AF0   IDX8   0842   IDX9   084A   IDXR   084C   IDXRE1 08EB
  IDXREG 0806   INM    0019   INM2   0036   INBUFF CD18   INH    0C35
  INH1   0C90   INH10  0CF8   INH11  0001   INH12  0D0E   INH13  0014
  INH14  0CA2   INH15  0E52   INH16  0CA9   INH17  0CB2   INH18  0CC1
  INH19  0CC7   INH2   0C9A   INH20  0CCD   INH21  0CDC   INH22  0CE8
  INH3   0C98   INH4   0C53   INH4B  0C65   INH5   0C96   INH6   0C8C
  INH7   0CEE   INH9   0006   LASTQ  0025   LINENO 0013   LK10E  12AA
  LK11E  12B3   LNBFPT CC14   LOOK10 12B4   LOOK11 12AA   LSTBLK 0027
  MEMEND CC2B   MODE   0030   MOVEN1 1882   MOVEN2 1B79   MOVENM 1B73
  MRIDX  0041   MSEC   003F   MSG2   11C7   MSG4   11E4   MSG5   1C16
  MSG6   1C56   NAMS   125D   NEXT   001E   NEXTT  0F66   NEXTT1 0F6A
  NEXTT2 0F70   NONCTG 0022   NXTCH  CD27   NXTOP  0019   OPTS   127B
  ORGS   1260   OUTADR CD45   OUTCH  CD0F   OUTCH2 CD12   OUTHEX CD3C
  OUTSW  CC22   P1A    0481   P1B    0450   P1C    0488   P1D    044F
  P1E    04BF   P1F    0479   P1G    04BC   P1H    04C2   P1I    04C5
  P1J    04C9   P1K    04EF   P1L    04DE   P2A    0589   P2B    04EE
  P2C    0519   P2D    0529   P2E    0552   P2E1   0556   P2F    0562
  P2F1   056C   P2F2   057F   P2G    059A   P2TAG  003D   P3A    060A
  P3C    06A6   P3D    05CA   P3E    05D7   P3F    0602   P3G    061A
  P3H    05E8   P4A    0627   P4AA   08C0   P4B    0637   P4C    063D
  P4C1   0653   P4D    0671   P4E    0688   P4END  0805   P4END1 08E4
  P4END2 08F3   P4END3 0919   P4END4 0925   P4F    0699   P4G    069E
  P4H    06DA   P4I    06ED   P4J    06F1   P4K    0700   P4K1   0720
  P4L    074A   P4M    076D   P4M1   0773   P4M2   07B0   P4M3   07D4
  P4M4   07B9   P4M5   07CD   P4M6   0798   P4N    07D8   P4O    07FF
  P4O1   0800   P4P    0817   P4Q    0868   P4R    0872   P4S    087A
  P4T    087F   P4U    0889   P4V    0892   P4W    0894   P4W1   08A6
  P4X    08AD   P4Y    08B6   P4Z    088E   PADDR  0050   PAGE   107B
  PAGE1  1087   PASCII 0088   PASS1  043A   PASS2  04E3   PASS3  059D
  PASS4  0618   PCOM   008E   PCRLF  CD24   PDEPTH 0000   PGMEND 1C77
  PHEAD  104A   PHEX   0055   PLINE  004A   PLNO   004A   POPCD  0069
  POPND  006F   PRE1   1CB7   PRE10  1CC3   PRE2   1CAD   PRE3   1D4F
  PRE4   1D0F   PRE5   1D39   PRE6   1D1F   PRE7   1D31   PRE8   1D4C
  PRE9   1CE9   PRELIM 1C77   PRINT  1012   PRINT3 1041   PSPL   120C
  PSTRNG CD1E   PTAG   0063   PUTCHR CD18   READ   007D   READ1  00C0
  READ2  00D0   READ3  00DC   READ4  00E4   READ5  00EE   READ6  0093
  READ7  00F0   READ8  00EC   READ9  00B5   READR  00F8   READR1 0E13
  READR2 0E1F   READR3 0E1C   REL    0C1B   REL1   0C2B   REL2   0C2D
  REPAS1 0997   REPAS2 0977   REPAS4 093A   REPAS6 0960   REPASS 0926
  REWIN1 0F63   REWIN2 0F2E   REWIND 0F16   RIDX   0023   RIP1   0ECC
  RIP3   0EE6   RIP5   0EF2   RIP6   0EF8   RIP8   0ED9   RIP9   0F00
  RIPPLE 0EC7   RPTERR CD3F   RRIDX  0018   SAVCMD 0029   SETEXT CD33
  START  0399   START1 03A3   STAT   0001   STRSEC 0011   TAGBLD 0032
  TAGCL1 1B8E   TAGCL2 1BA1   TAGCL3 1BA5   TAGCLN 1B8A   TAGEND 0009
  TAGST  0007   TENCMD 1983   TFIG   11FB   TRIP4  0E97   TRIP5  0E8F
```

```
TRIPPL 0E72   TTGEND 0000   TTGST  0008   TTYDEP 0049   TTYEOL C002
WARNS  C003   WRTDS1 0FDC   WRTDS2 0FF1   WRTDS3 0FFF   WRTDS4 1004
WRTDS5 0FE4   WRTDSC 0FD5   WRTERR 0BE0   XFER   0024   XFRADD 000F
XREND  0044   XRFI10 10F8   XRFI11 1118   XRFIN1 1088   XRFIN2 10C7
XRFIN3 1003   XRFIN4 1008   XRFIN6 10E7   XRFIN7 10EB   XRFIN8 10FC
XRFIN9 1104   XRFLAG 0010   XRPLN  1193   XRPLN1 11AA   XRPLN2 11B6
XRPLN4 11B8   XRPRN1 1130   XRPRN2 1170   XRPRN3 1181   XRPRN4 1188
XRPRN5 118F   XRPRN6 112E   XRPRN7 1145   XRPRNT 1126   XRSOR1 10AF
XRSOR9 1124   XRSORT 109E   XRST   0042   XRTAG  0046   ZAPFLG 0020
ZAPT   0E99   ZAPT2  0EAC   ZAPT3  0EB0   ZAPT4  0EAE   ZAPT5  0BC2
ZAPT6  0BC3   ZSUPP  1091   ZSUPP1 1090   ZZZZY  1983   ZZZZZ  1283
```

BIT BUCKET

SWTPc - Prospering Pioneer

Microcomputers. See that word and you think of those ubiquitous marvels of technology that represent the miracle of modern electronics in the home and workplace. With the variety of machines available today, it is difficult to visualize a world that does not include these machines. But there was a time when there were no microcomputers on the market at all. In the early 1970's, if computing was done, it was done on mainframes or minicomputers. That is, until 1975...

Southwest Technical Products Corporation, San Antonio, Texas, began manufacturing microcomputer systems that year with kit systems based on the Motorola 6800. With RAM capacity of 32K, these systems provided hobbyists with the ability to have computer systems of their own, with considerable power, at a reasonable price. Running an 8K, cassette-loaded BASIC, these early systems brought computers into the home for the first time.

BEGINNINGS 1975 was the beginning of SWTPc's involvement with selling microcomputers, but the company had begun some eleven years earlier with kit electronics produced in the garage of entrepreneur Daniel Meyer. Electronics hobbyists will remember the product line of amplifiers, graphic equalizers, color organs, digital meters, strobe lights, ultrasonic burglar alarms, timers, counting units, the Beachcomber metal detector and the Theremin, an innovative electronic musical instrument. It was as a natural extension of this varied product line that Southwest took its first steps into microcomputers.

Microprocessor - based machines were introduced at about the same time by SWTPc, MITS (with the Altair 8800) and IMS Associates (with the IMSAI 8080). For the 6800 computer, Southwest invented the SS-50 bus, with 7 slots for processor, memory and controllers, and including a separate 30-pin I/O bus with 8 slots on the motherboard. The system ran at a clock speed of 1.0 MHz. Auxiliary storage was accomplished on this early system through audio cassette player/recorder.

DISK OPERATING SYSTEM The first disk operating system for the 6800 computer was introduced to the micro world in 1977. Running on floppys, FLEX was developed by Technical Systems Consultants, now of Chapel Hill, North Carolina. Designed to be powerful and user-friendly, FLEX became the defacto standard operating system for the 6800.

6809 COMPUTERS In 1978, Southwest introduced the /09 computer based on the new Motorola 6809 microprocessor. This improved processor gave the system user a greatly enlarged instruction set, as compared with the 6800, and provided the programmer with four pointer registers and a program counter... each of 16 bits. Of the Motorola models, Southwest chose to use the 68B09, for 2.0 MHz operation. For users wishing to upgrade their 6800 systems, Southwest provided update documentation to accomodate the new processor.

MULTI-USER SYSTEMS With the power of the 6809 processor now available, the decision was made to implement a multi-user operating system for the SWTPc computer. After close examination of the available OS designs, SWTPc chose UniFLEX*... a multi-user

OS closely modeled on the Bell Labs' UNIX** system. Also implemented by TSC, UniFLEX incorporated some of the user - friendly attributes of FLEX.

S/09 AND S+ SYSTEMS The multi-user system, dubbed the S/09, was offered for the first time in 1979. This system provided features previously expected only from minicomputers and mainframes... Multi-terminal (up to 12) operation, Dynamic Memory Allocation and Direct Memory Access. Furthermore, the S/09 system was configurable to include "Winchester" hard disk drive technology. With the addition of a hard disk, the S/09 achieved full status as a new, hybrid computer classification... the MICRO-MINI.

The state-of-the-art in SWTPc micro-minis is the S+ system. Designed for office, school lab and industrial data processing, the S+ provides for operation of up to 32 terminals through the use of effecient data handling, larger memory capability and peripheral processor interface. Running in a supervisor and user state, the S+ CPU monitors system functions to prevent user - generated system crashes. The S+ can also run the MUMPS Operating system with up to 16 Terminals.

Both the S+ and the S/09 support 14" and 5-1/4" Winchester disk drives, logical tape backup, and up to 5Mbytes of floppy disk storage. A variety of letter-quality and dot-matrix printers are available for use with either system.

X12+ AND BEYOND... Southwest Technical Products continues to develop and manufacture microcomputer systems to meet the increasing data processing needs of the marketplace.

The new X12+ is a desk-top, 256K computer integrated with a SWTPc-manufactured intelligent terminal. Running UniFLEX, the system features Mini-Winchester auxiliary storage, data communications, optional hardware floating point computation and supports 2 additional work stations.

Other hardware under development at the SWTPc 5-building facility in San Antonio includes plug-in 68000 processor, floating point processor, virtual system processor, 1 Mbyte memory board and networking interfaces.

SUMMARY In the fast-paced and volatile world of microcomputer manufacturing, it takes innovative engineering and aggressive pricing to stay in competition. Since the very birth of the micro, Southwest Technical Products has provided this kind of leadership in design and production of computer systems. Southwest Technical Products is the Prospering Pioneer of microcomputers.

* UniFLEX and is a trademark of Technical Systems Consultants ** UNIX is a trademark of Bell Labs.

The following is a description of the X12+ system from SWTPc:

X-12+ Multi-user computer system

Concept:

The X-12+ system is designed as a stand-alone 256K computer capable of supporting one to three users simultaneously. The system incorporates as standard features many capabilities that micro users are coming to expect from their systems--hard disk capacity; multi-user, multi-tasking, multi-processing capability; expandability to 1024K RAM; variety of available programming languages and proven operating systems.

The Hardware:

The X-12+ CPU resides inside the SWTPc X12 terminal, with a 12 inch (diagonal) CRT and a detached 91-key keyboard. Standard auxiliary storage sits alongside the CRT in an attractive "Mailbox Memory" unit that contains a high speed 20M-byte mini Winchester and a 1.25M-byte mini floppy drive.

System architecture is built around the Motorola 68809 processor, and includes extended addressing, optional TMS 320 32-bit processor for floating point work (transparent to the user), 256K of dynamic RAM, buffered disk I/O to reduce redundant disk reads, two RS-232 ports for additional terminals, and one parallel port for future communications with loop network. The X-12+ communicates with it's own CRT in parallel for ultra fast output display. One RS-232 printer port and one 'Centronics Parallel' printer port are included as standard. An additional RS-232 serial port is included to allow the X-12+ to act as a terminal on a remote system.

CRT Features

The X-12+ display is controlled by a separate 6809 processor, and an on-board vocabulary of 255 words may be spoken by the voice processor installed as standard equipment. User-programmable character sets, multiple screen formats, 15 programmable function keys, tone generator, reverse fielding, under and over-lining, blinking characters, field protection, multiple screen formats, display status line, graphics and ergonomic design are additional features of the unit.

Software:

The standard X-12+ will run the powerful *UniFLEX Operating System from Technical Systems Consultants. UniFLEX is optimized for the 6809, and provides the same file structure and 'shell' facility that is available with **UNIX. UniFLEX dynamically allocates up to 64K of memory for each running task in 4K increments, and supports BASIC, FORTRAN, Pascal, 'C', COBOL, a relocating assembler and Pilot. Input/Output redirection, piping, random and sequential files and virtual array capability provide for the complete range of file access needs.

As an additional option, the X-12+ will be configurable to run the single-user FLEX+ operating system, with dual-floppy operation.

Special Feature:

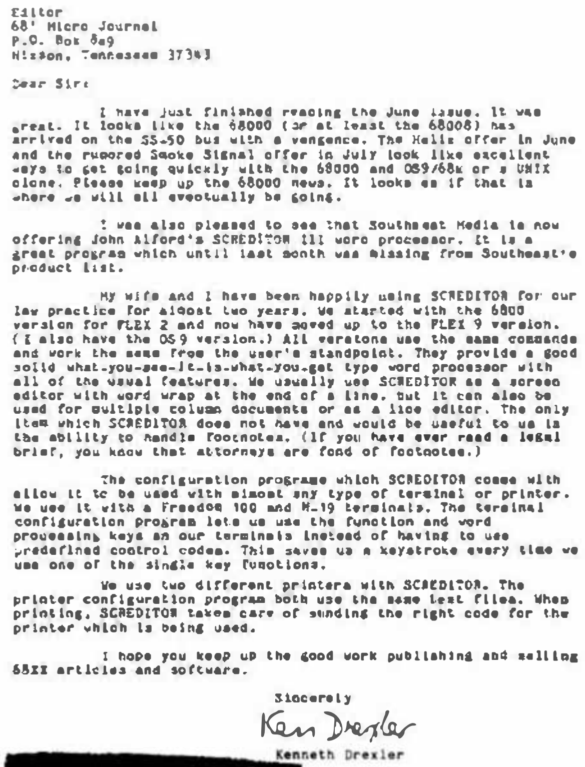
The X-12+ has capability to run in Local and Remote modes. The unit can function as an intelligent terminal on another system while two other terminals attached to it are using the local system resources. Future development will include synchronous loop network capability for up to 255 X-12+ units, each with up to 2 additional workstations.

Pricing: The X-12+ multi-user configuration of CPU, 256K RAM, CRT, detached Keyboard, Voice and Sound synthesizers, three parallel ports, four serial ports, 1.25M floppy, 20M mini- Winchester and UniFLEX Operating System ... $6,595.

Single-user configuration with dual 1.25M floppies, FLEX+ Operating system, full CRT features and BASIC... $4,495.

*UniFLEX is a trademark of Technical Systems Consultants, Inc.
**UNIX is a trademark of Bell Laboratories.

- - -



Editor
68' Micro Journal
P.O. Box 849
Hixson, Tennessee 37343

Dear Sir:

I have just finished reading the June issue. It was great. It looks like the 68000 (or at least the 68008) has arrived on the SS-50 bus with a vengence. The Helix offer in June and the rumored Smoke Signal offer in July look like excellent ways to get going quickly with the 68000 and OS9/68K or a UNIX clone. Please keep up the 68000 news. It looks as if that is where we will all eventually be going.

I was also pleased to see that Southeast Media is now offering John Alford's SCREDITOR III word processor. It is a great program which until last month was missing from Southeast's product list.

My wife and I have been happily using SCREDITOR for our law practice for almost two years. We started with the 6800 version for FLEX 2 and now have moved up to the FLEX 9 version. (I also have the OS9 version.) All versions use the same commands and work the same from the user's standpoint. They provide a good solid what-you-see-is-what-you-get type word processor with all of the usual features. We usually use SCREDITOR as a screen editor with word wrap at the end of a line, but it can also be used for multiple column documents or as a line editor. The only item which SCREDITOR does not have and would be useful to us is the ability to handle footnotes. (If you have ever read a legal brief, you know that attorneys are fond of footnotes.)

The configuration programs which SCREDITOR comes with allow it to be used with almost any type of terminal or printer. We use it with a Freedom 100 and H-19 terminals. The terminal configuration program lets us use the function and word processing keys on our terminals instead of having to use predefined control codes. This saves us a keystroke every time we use one of the single key functions.

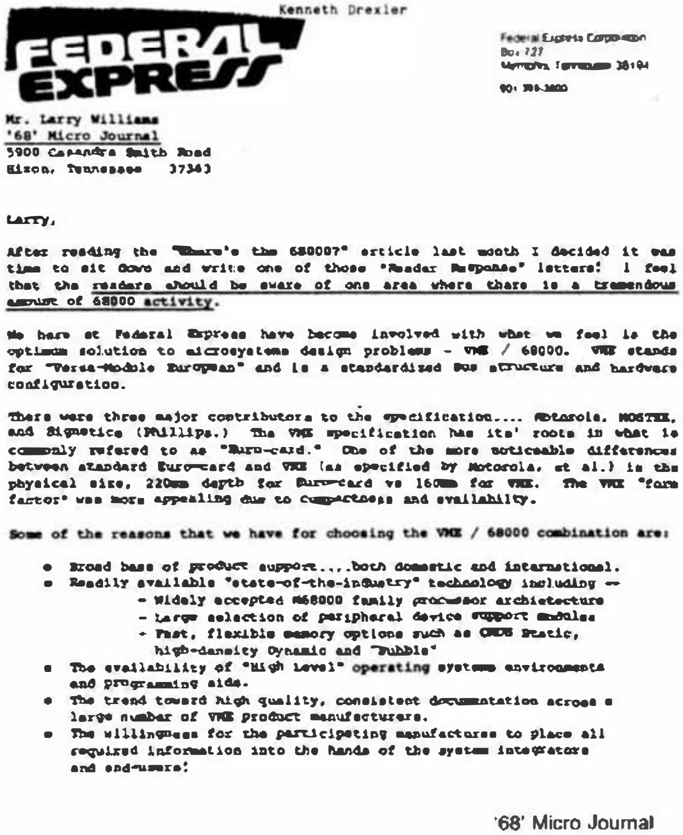
We use two different printers with SCREDITOR. The printer configuration program both use the same text files. When printing, SCREDITOR takes care of sending the right code for the printer which is being used.

I hope you keep up the good work publishing and selling 68XX articles and software.

Sincerely

Ken Drexler

Kenneth Drexler



FEDERAL EXPRESS

Federal Express Corporation
Box 727
Memphis, Tennessee 38194

901 369-3600

Mr. Larry Williams
'68' Micro Journal
5900 Cassandra Smith Road
Hixson, Tennessee 37343

Larry,

After reading the "Where's the 68000?" article last month I decided it was time to sit down and write one of those "Reader Response" letters! I feel that the readers should be aware of one area where there is a tremendous amount of 68000 activity.

We here at Federal Express have become involved with what we feel is the optimum solution to microsystems design problems - VME / 68000. VME stands for "Versa-Module European" and is a standardized bus structure and hardware configuration.

There were three major contributors to the specification.... Motorola, MOSTEK, and Signetics (Phillips.) The VME specification has its' roots in what is commonly refered to as "Euro-card." One of the more noticeable differences between standard Euro-card and VME (as specified by Motorola, et al.) is the physical size, 220mm depth for Euro-card vs 160mm for VME. The VME "form factor" was more appealing due to compactness and availabilty.

Some of the reasons that we have for choosing the VME / 68000 combination are:

- Broad base of product support....both domestic and international.
- Readily available "state-of-the-industry" technology including --
    - Widely accepted M68000 family processor archietecture
    - Large selection of peripheral device support modules
    - Fast, flexible memory options such as CMOS Static, high-density Dynamic and "Bubble"
- The availability of "High Level" operating systems environments and programming aids.
- The trend toward high quality, consistent documentation across a large number of VME product manufacturers.
- The willingness for the participating manufactures to place all required information into the hands of the system integrators and end-users!

We are very comfortable with the direction we have chosen. A tremendous amount of effort was put into the process. The results have been both successful and rewarding. Finally, we are looking forward to a long and productive involvment with VME / 68000 based products.

Kenneth R. Lewis
MICROSYSTEMS

MICRO-MATION
INCORPORATED
801 Duchess Road
Bothell, WA 98012
(206) 481-2812

Computer Publishing Center
68 Micro Journal
P.O. Box 849
Hixson, TN 37343

attn: Don Williams & staff

Dear Don,

I haven't had time to write many letters the past couple of years but I still find time to read your magazine. I've looked at a couple of the competitors' but they don't seem to even finish out my years subscription before disappearing. I've recently solved a small problem that has been nagging me for a couple of years on Flex based systems. I've owned a couple of systems that provide a ready input to the floppy disk controller, either from the drives themselves or from an index pulse detector, with the idea that if a drive isn't ready then trying to access it will result in an error message rather than a hung system. Somehow it never seemed to work quite right and strange things seemed to happen, such as getting a "NOT FOUND" message when trying to use a system utility that was definitely present. I don't want to come out & say that there is a bug, either in Flex or in the documentation, because I might be the only one that has had problems.

First, let me state that I have written my own disk drivers, so the problem might be mine alone. However, I should also state that those drivers are similar to and patterned after those suggested by TSC in the Flex Adaptation Manual for General Flex. Now that I've incuded all these disclaimers, let me tell you how the problem occurs, and one possible solution. The Western Digital series of floppy disk controller chips (1771,1971,2791) have a ready input which can be used to detect if the drive is ready for access (motor on, speed up, disk in place, etc.) What sometimes happens is that the disk operating system (DOS) issues a read sector command (say for the directory) and the controller attempts to read the sector called for; however, if the drive isn't ready the controller chip will immediately abort the operation and return with the "not ready" flag set. If the READ sector subroutine doesn't test this bit from the controller, and the example provided by TSC doesn't, then the DOS thinks that a valid read has occurred without any data whatsoever being read from the disk. NOW THAT CAN CAUSE SOME STRANGE RESULTS !!

Two possible fixes suggest themselves, one is to add the ready bit to the read sector error test; the other is to perform a drive ready test before trying to read the sector. The latter is what I chose to do, adding a 1/2 second delay & retest if not ready. The same addition was made to the write sector routine.

Very truly yours,

Joe Wicklund

651 Pammel Court
Ames, Iowa 50010
June 3, 1984

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith Rd.
P.O. Box 849
Hixson, TN 37343

Dear Sirs:

In response to the letter of Mr. LeFarr Stuart which appeared in your June issue, I would like to ask Mr. Stuart, why, if he truly believes that each computer system has its good and bad points, does he make the totally negative statement calling the Color Computer a "useless toy! inferior to the Commodore 64"?

Well, I am the owner of a 64K Color Computer running OS-9 and although it is not my idea of the ideal computer system it is far from useless. I have had my complaints with the machine, the limited memory, the bad keyboard and, yes, the useless display but have solved many of these problems to an extent myself and it seems that Tandy too is solving some of them. Aside from that the CoCo has several very good points. For starters, where else can you get a 6809 based machine running OS-9 or FLEX for under $1000? As a computer science major at Iowa State University I have found my CoCo a very valuable computer. For one thing it has given me an opportunity to use and study an advanced UNIX-like operating system on my "toy" computer. Secondly, it has given me a chance to program in assembler and high level languages at home, not just in classwork. For example, the C language is not currently taught to undergraduates at ISU but with my CoCo and Micorware's C compiler I now have a chance to learn C at home and with the full compiler I can write usefull, transportable code which I could not do with other micros and their Small C's. Aside from C, I can program in BASIC09 which I vastly prefer over Microsoft BASIC or any of the other BASIC dialects on the market, once you are used to modular programming it is hard to go back to using GOTO's and GOSUB's. Within a few months Microware's Pascal should also be on the market giving another full Pascal compiler for the CoCo, one exists for RS-DOS and another for FLEX already. What other low cost micro on the market has all these full languages available to it and under several different operating systems?

Thus I see the Color Computer as being far from a "useless toy" and I also see it as far superior to the Commodore 64 and all the other low cost micros on the market. My views are not singular either, the CoCo has a large following as can be seen by the large number of clubs, magazines and software suppliers who support it. I know alot of the software is games but there is getting to be more serious software out now too from FHL, Computerware, JBM, SouthEast Media and Tandy's Microware's releases. Besides it is nice to sit back and play a game now and then. If I was running a buisiness it would be nice to have a S-50 bus 6809 or 68000 machine but for me at home the CoCo is just right and has been a very useful "toy".

Sincerely,

Tim Harris

Tim Harris

A while back I needed a terminal for my "Mikbug Board" (G-Job) to be able to use the thing at home. I was always using a terminal at work. Now with a little work I have a "semi-intelligent" terminal. The terminal uses the Motorola family of chips, ie a 6802 processor, 6847 video generator, and 6821 for input/output. Since I was using the 6847 in something like "TVBUG" I had to wait for some kind of sync pulse to load or read the display ram. Now the 6802 is in a Interleaved Direct Memory Access mode. The 6802 is dedicated to the 6847 in a terminal mode. There used to be snow when the 6802 talked to the 6847, now there is no, I repeat no snow when display ram is accessed! This is done by syncing the 6802 to the 6847 at power-up with a either-edge triggered device. Together with a 8 bit latch, the 6802 and 6847 share the same memory and they do not know about each other (kind of like the Radio Shack 6883 setup). The reason for this modification is for the 8 point move, draw, box, rectangle and crosshair routines. Originally the routines took a lot of time to draw any line because of the waiting for the sync pulse to do memory accesses. Finally the full speed of these routines can be used to an advantage. Yes the processor runs at color-burst crystal speed, but I can tell you that the graphics routines are quite fast. Also the "TVBUG" can be modified to accept these routines (and hardware mod) without having to use the handshaking for blanking sync thereby increasing throughput and eliminating any snow on the screen in any graphics mode. So, now I have a semi intelligent terminal that runs at processor speed as the input/output is 8 bit parallel handshaking. The firm ware is just under 2K (hex) bytes of data, including the graphics routines. If you would like the schematics and firmware, please contact me about these items. I will be glad to make these public.

After this terminal was built, I wanted to run 6800 type programs a little faster so I upped my 6800 to a 6809. Currently I am running Santa Barbara Tiny Basic with extensions (poke, peek, wait and various terminal oriented functions).

All in all I have a delightful combination to use. This is only the beginning of course, there is always FLEX to add in someday.

Paul T. Barton
1412 N.E. 46th. Ave.
Portland, Oregon
97213

Analog Micro Systems
3660 Valmont Road
Boulder, Colorado 80301
(303)444-6809

January 1984

V1.3

News Release

Robot-1

With the advent of large scale integrated circuitry (LSI), the availability of powerful home personal computers has become a reality. This same technology has also shown up in industry to further automate dangerous or tedious and precision assembly procedures in mass production lines around the world.

Now, for the first time in history the availability of this technology has made possible the introduction of low cost home personal robotics and low cost industrial robotic systems. A vast interest in artificial intelligence and robotics has arisen due to the availability of the small powerful computer systems that make intelligent robot systems possible.

As an entrance into the field of robotics Analog Micro Systems is offering a complete line of computer controlled robotics for entertainment, education, and small industry. The first product to be released for national sale is the "Robot-1" computer servo controlled robot arm. Two versions of this robot arm are now being offered on the general market. One version is a plug in compatible output device for the Radio Shack Color Computer. The "Robot-1" is controlled by either keyboard or joystick inputs to the color computer. The operator can put the computer in a teach mode and operate the robot arm to perform a given task. After the operator has guided the Robot-1 through the various movements and functions, the operator may then instruct the computer to remember and repeat the sequence of movements and functions any number of times. The movement sequences may also be saved on tape or disk for later use. The Robot-1C comes complete with all software and includes a power supply, a six channel servo controller, all cables and a Robot-1. The Robot-1C is priced at $395.00 complete.

The next version of Robot-1 comes with different interface electronics in order to function on a larger computer system which like the Color Computer uses the 6809 microprocessor. This will allow Color Computer users to upgrade their applications to multi robot systems while using the same software that they developed for the color computer. This minicomputer like system is commonly known as the SS-50 buss. The Robot-1S uses the sixteen address thirty pin input-output connector on the SS-50 buss to control a six axis servo controller and power supply interface card. The Robot-1 takes advantage of the larger computer buss by allowing up to eight robot arms to run interactively and simultaneously using the AMS multi robot arm software. All movement sequences may be saved in memory or on disk to be retrieved later repeating the multi interactive robot arm movement sequences. Source code is provided with all robot software to enhance the ease of research and experimentation in the field of robotics.

The techniques that are learned from using Robot-1 may be directly transferred to the use of Robot-2 and Robot-3 industrial work robot arms as the software techniques, the control techniques, and the interface hardware are identical. Robot-1 is aimed at robotic entertainment, research, and education. Robot-2 is useful for small industrial assembly such as electronic circuit boards. Robot-3 is a larger version used for medium duty operations such as chassis assembly, painting operations, or the handling of dangerous materials. All three robot arms plug into the same interface electronics allowing low cost upgrade to industrial applications. Additional software will be supplied with each robot arm.

A number of accessories shall be available for Robot-1 and it's successors. Software packages shall be available to move Robot-1 from the input of AMS biofeedback equipment allowing certain brainwave patterns, muscle movements, and body functions to control the robot arm movements. These are available for the Analog Micro Systems Model 8266 Electroencephalograph Brainwave Analyzer, the Model 8264 Biofeedback Precision Temperature Monitor, the Model 8262 Electromyograph, and the Model 8260 Dermograph. A low cost digital video TV camera shall be available for robotic sight and pattern recognition. An ultrasonic range finder shall be available for distance measuring. A mobile cart to fit the robot arm base into will give the robot arm mobility. Should cables to the robot arm be undesirable a radio link shall be available for remote control of the mobile based robot arm. Computer generated voice output along with sound recognition is available directly to the computer as well as over the radio link to the mobile robot system. A four joystick and switch interface is available for computers not having such inputs. The above accessories shall be available for SS-50 based computer systems.

The robotics systems outlined above are modular in nature and may be assembled in any number of possible configurations. For example, a remote radio controlled mobile robot with speech, sound recognition, ultra sonic ranging and detection, and robot vision may be assembled with the above modular components. The resulting system can be programmed in a high level computer language resulting in a superior artificial intelligence system with a minimum of human effort. The performance to cost ratio of the above systems far exceeds anything that is available on today's market.

Crummy CoCo

Lane P. Lester, Ph.D.
Professor of Biology
Liberty Baptist College
Lynchburg, VA 24506

In his letter printed in the June issue of *68 Micro Journal*, LaFarr Stuart calls the Color Computer a "useless toy", and says that "nobody with a grain of sense would every (sic) consider the Color Computer for anything serious". It's fortunate for me that I lack a grain of sense, because the following are some of the things I do with my "useless toy".

1. Word processing: Lecture notes, tests, correspondence, articles for magazines, and laboratory manuals.

2. List management: lists of several hundred 35mm slides, supplies to be ordered, addresses, and phone numbers are maintained. Selecting and sorting functions allow reporting in selected formats.

3. Scheduling: All appointments, lecture and test schedules, and special dates are kept in a single file which can be searched in order to print specialized schedules.

4. Grade maintenance: Multiple choice tests are graded and questions analyzed for quality. Test scores are stored and assignment scores are entered. Functions allow points to be converted to letter grades.

5. Income tax management: quarterly evaluations of tax liability minimize withholding and the chore at year's end. Multiple forms are provided.

6. Lecture graphics: certain topics in biology lend themselves to graphic presentation in color and with animation.

7. Simulations: My genetics students use a program with which each student simulates animal crosses. The offspring can be used in further crosses in order to determine the type of inheritance involved in the various traits.

No one claims that the CoCo is the equivalent of a Gimix, but for $200 a beginner in personal computing can purchase a 64K 6809 machine with Extended BASIC, cassette interface, TV interface, RS-232 port, and two A/D converter ports. If his interest wanes, his doorstop hasn't cost him a fortune. If like mine his interest grows, he has a machine that can grow with him.

If Disk BASIC ceases to satisfy for whatever reason, there is FLEX ("bastardized" though it may be... LaFarr's term, not mine) and OS-9 (although its appropriateness on the CoCo can be questioned). These additional DOSes open great vistas of programming in additional languages (PL/9 is my current fascination). And expansion interfaces allow the use of the types of boards which SS-50 users enjoy: 80x25 video displays, serial and parallel ports, clock/calendars, static memory, and stereo music synthesis.

Ron Anderson says that the small size of the SS-50 software market may eventually kill the bus. The "useless toy" has introduced many thousands to the pleasure and power of 6809 computing. If the efforts of software producers in "getting it to go on the CoCo" (LaFarr) are any indication of the potential market, the Color Computer may postpone the death of the SS-50.

Dear Don,

Getting out of XBASIC

There's a much easier way:

```
29999 EXEC. "exec,flex.txt"
```

And "flex.txt" just reads "asn" or anything. After executing "flex.txt", exec.cmd returns to FLEX.

Yours sincerely

Hans-Josef Heck

Breslauerstr.25
5630 Remscheid
Western Germany

May 8, 1984
1201 Bradley Rd.
Lynden, WA 98264

"68" Micro Journal

Dir Sirs:

The folling program is an extension of the os-9 command, procs e. What this program does is take certain information from a procs e. Then it compiles this information into an easy to read format.

First it will take the user name from the procs, then it will look at what they are doing. After it has taken this information it will dump it onto the screen, giving the name of the user and what they are doing at the time you execute the program.

I suggest that this program be packed and put on the cmds directory so it can be executed from os-9.

Before you can run the program, you have to set up a data file on the sys directory. I have also written a program to do this. What it does is to take the user number and name from the password word file and put into a file of its own.

This program makes it easier to monitor the activies of other users on the system, making it easier to maintain security (Which by the way was what it was originally designed for).

Sincerely

Michael W Klein
Systems Manager
Lynden High School

```
PROCEDURE procsu_fancy
 0000        REM ****************************************
 0029        REM *                                      *
 0052        REM *  Security Program Designed and       *
 0078        REM *  Written at Lynden High School by    *
 00A4        REM *  Michael W. Klein on May _, 1984.    *
 00CD        REM *                                      *
 00F6        REM ****************************************
 011F        DIM path:BYTE
 0126        DIM line:STRING[80]
 0132        DIM num,procs,proc(50):STRING
 0146        DIM count,number(50):INTEGER
 0156        SHELL "procs e )/h0/sys/proc_dat"
 0173        OPEN #path,"/h0/sys/proc_dat":READ
 018E        FOR i=1 TO 4
 01A0          READ #path,line
 01AA        NEXT i
 01B5        count=0
 01BC        WHILE NOT(EOF(#path)) DO
 01C7          count=count+1
 01D2          READ #path,line
 01DC          num=MID$(line,10,4)
 01E9          procs=MID$(line,43,16)
 01F6          proc(count)=procs
 0202          number(count)=VAL(num)
 0210        ENDWHILE
 0214        CLOSE #path
 021A        SHELL "del /h0/sys/proc-dat"
 0232        DIM name,password_number:STRING
 023D        OPEN #path,"/h0/sys/user_no.":READ
 0258        PRINT "Date:   "; DATE$
 0265        PRINT "User Name","Process"
 027C        PRINT "------------------------------"
 029D        PRINT
 029F        WHILE NOT(EOF(#path)) DO
 02AA          READ #path,name,password_number
 02B8          FOR j=1 TO count
 02CD            IF VAL(password_number)=number(j) THEN
 02DE              PRINT name,proc(j)
 02EB            ENDIF
 02ED          NEXT j
 02F8        ENDWHILE
 02FC        CLOSE #path

PROCEDURE user_num_fancy
 0000        DIM path,p:BYTE
 0008        DIM line:STRING[80]
 0017        DIM name,number:STRING
 0022        OPEN #path,"password":READ
 0035        CREATE #p,"user_no.":WRITE
 0048        WHILE NOT(EOF(#path)) DO
 0053          READ #path,line
 005D          x=SUBSTR(",",line)
 006A          name=LEFT$(line,x_1)
 0078          line=RIGHT$(line,LEN(line)-x)
 008E          x=SUBSTR(",",line)
 0098          line=RIGHT$(line,LEN(line)-x)
 00AE          z=SUBSTR(",",line)
 00BE          number=LEFT$(line,z-1)
 00CC          WRITE #p,name,number
 00DA        ENDWHILE
 00DE        CLOSE #path
 00E4        CLOSE #p
```

HYPERDISK

```
 OPT PAG
 TTL HELIX DISK SIMULATOR
 PAG
* HYPERDISK -- IN MEMORY DISK SIMULATOR FOR HELIX COMPUTER
*   UTILIZES EXPANDED MEMORY TO SIMULATE A DISK DRIVE

* COPYRIGHT 1982 COMPUTER SYSTEMS CENTER OF HAZELWOOD, INC.

* WHEN EXECUTED AS A COMMAND EXPANDED MEMORY IS SEARCHED
*  OUT AND FORMATTED AS A DISK,  DISK I/O CALLS ARE THEN
*  INTERCEPTED AND ALL REFERENCES TO DISK 3 ARE DIRECTED
*  TO THE SIMULATED DISK, ALL OTHERS ARE PASSED ON TO THE
*  PHYSICAL DISK DRIVERS.

* METHOD:
*   THE 'E' DAT REGISTER IS USED AS A WINDOW TO THE EXTENDED
*   MEMORY.  THIS ASSURES THERE WILL NO CONFLICT WITH ANY
*   SYSTEM REFERENCES SINCE 'E' IS NORMALLY RESERVED FOR I/O
*   AND DURING DISK SIMULATION IT IS THE I/O.
*   THE SIMULATED DISK IS 'FORMATTED' INTO 'TRACKS' OF 16
*    SECTORS' WHICH IS EXACTLY 4096 BYTES, THE SIZE OF A DAT
*   PAGE.  THE TRACK/SECTOR LINKS ARE SET UP SO THAT THE
*    'TRACK' IS THE ACTUAL VALUE TO STUFF INTO THE DAT AND THE
*    'SECTOR' IS THE HIGH ADDRESS BYTE OF THE BLOCK TO REFERENCE.
*   THE 'TRACKS' ARE FORMATTED WITH NON-CONTIGUOUS LINKS IN ORDER
*   TO SKIP OVER THE 'E' WINDOW ADDRESS AND THE UNMAPPABLE 'F'
*   ADDRESSES.  THE 'SECTOR' ADDRESS IS ACTUALLY ONE RORE THAN
*   THE ACTUAL MEMORY ADDRESS BYTE IN CASE SOME PROGRAM ASSUMES
*   THAN SECTOR ADDRESSES CAN NEVER BE ZERO OR THAT A SECTOR LINK
*   ADDRESS OF ZERO IS AS GOOD AS TRK/SEC OF ZERO TO INDICATE
*   END OF FILE.
*   THE ABSOLUTE DISK TRK/SEC OF 0003 (STR) AND 0005 (DIR START)
*   ARE MAPPED INTO SIMULATED 0101 AND 0102. 'TRACKS' 01 AND 02
*   ARE FORMATTED AS THE DIRECTORY GIVING 31 SECTORS WHICH IS
*   APPROXIMATELY THE SAME RATIO AS ON ACTUAL DISKS.
 PAS
* SYSTEM EQUATES

WARMS EQU $CD03 RETURN TO SYSTEM
PSTRNG EQU $CD1E PRINT STRING
NXTCH EQU $CD27 SET NEXT CHAR
OUTDEC EQU $CD39 OUTPUT DECIMAL NUM
INDEC EQU $CD48 INPUT DECIMAL FROM INBUF
TTYEOL EQU $CC02 COMMAND SEPARATOR
LSTERM EQU $CC11 LAST TERMINATOR
DATE EQU $CC0E SYSTEM DATE
DSKVEC EQU $DE00 DISK VECTOR TABLE
LBFPTR  EQU     $CC14   FLEX LINE BUFFER POINTER
OUTHEX  EQU     $CD3C   OUTPUT HEX NUMBER
PUTCHR  EQU     $CD18   PUT CHARACTER

* FIXED DISK TRK/SEC ADDRS

SIRSEC EQU $0003 SYSTEM INFORMATION RECORD
DIRSEC EQU $0005 DIRECTORY START

* HARDWARE REGISTERS

DAT EQU $FFFE DAT RESISTER USED AS WINDOW
WINDOW EQU $E000 BASE ADDR OF WINDOW REFERENCE
STDDAT EQU $FE 'STANDARD' DAT SETTING

* CHARACTER EQUATES

CR EQU $D CARRIAGE RETURN
EOL EQU 4 END OF LINE MARKER
 PAG
 ORG $C100 RUN FORMATTER IN UTILITY AREA

* SIZE AND 'FORMAT' MEMORY

START LDA #3 SET DEFAULT DRIVE NUM
 STA DRVNUM
        LDD     #$10FF
        STD     STBLCK  SET DEFAULT START AND END BLOCKS
PARSE4  CLR     STFLAG  CLEAR 'START BLOCK PROCESSED' FLAG
        CLR     DONFLG  CLEAR 'COMMAND LINE PROCESSED' FLAG
PARSE2  JSR     GETHEX
PARSE1  TSTB
        BEQ     NOTNUM  GO IF NUMBER NOT FOUND
        CMPB    #2
        BEQ     TWODIG  GO IF TWO DIGIT NUMBER FOUND
        CMPB    #5
        BEQ     TWODIG  5 DIGIT PROCESSING SAME AS 2 DIGIT
        CMPB    #1
        LBNE    SYNERR  SYNTAX ERROR IF NUMBER OTHER THAN 1, 2, OR 5 DIGITS
        CMPA    #9
        LBHI    SYNERR  SYNTAX ERROR IF NUMBER >9
        STA     DRVNUM  STORE DRIVE NUMBER
```

```
PARSE3  JSR     GETHEX  GET NEXT ITEM
        CMPB    #1      DON'T ALLOW ANOTHER 1 DIGIT NUMBER
        LBEQ    SYNERR
        BRA     PARSE3  CONTINUE PARSE
NOTNUM  LDA     TERMCH
        CMPA    #'-
        BEQ     DASH
        CMPA    #'+
        BEQ     OPTION
        CMPA    #CR
        LBEQ    ENDCMD
        CMPA    TTYEOL
        LBEQ    ENDCMD
        CMPA    #$20
        BEQ     PARSE2
        BRA     SYNERR
DASH    INC     STFLAG  SET 'START BLOCK PROCESSED' FLAG
        BRA     PARSE3  CONTINUE PARSE
TWODIG  TST     STFLAG  PROCESSED START BLOCK YET?
        BNE     SETEBK  YES, SET END BLOCK
        INC     STFLAG  SET FLAG NOW
        STA     STBLCK  SET START BLOCK
        LDA     TERMCH  GET TERMINATING CHARACTER
        CMPA    #'-     BETTER BE DASH
        BNE     SYNERR  ELSE SYNTAX ERROR
        BRA     PARSE3  CONTINUE PARSE
SETEBK  CMPA    STBLCK  MAKE SURE END>START
        BLO     SYNERR
        STA     ENDBLK
        LDA     TERMCH  SEE IF WE'RE DONE
        CMPA    #CR
        BEQ     ENDCMD
        CMPA    TTYEOL
        BEQ     ENDCMD
        CMPA    #$20    BETTER BE SPACE THEN
        BNE     SYNERR
        LDA     [LBFPTR] CHECK NEXT CHARACTER
        CMPA    #'+     OPTION?
        BNE     FORMAT  NO, MUST BE NEW DRIVE #
        JSR     NEXTCH  SKIP OVER +
OPTION  JSR     NEXTCH  GET OPTION
        PSHS    A       SAVE CHAR
        JSR     NEXTCH
        CMPA    #CR
        BEQ     SETECM  SET END OF COMMAND FLAG IF CR
        CMPA    TTYEOL  ALSO IF TTYEOL
        BEQ     SETECM
OCONT   PULS    A
        CMPA #'N
 LBEQ NOFMT DONT FORMAT MEMORY
 CMPA #'n
 LBEQ NOFMT
 CMPA #'R
 LBEQ REMOVE    REMOVE ALL SIMULATED DRIVES
 CMPA #'r
 LBEQ REMOVE
        CMPA    #'D
        LBEQ    DELETE  DELETE SPECIFIED DRIVE # ONLY
        CMPA    #'d
        LBEQ    DELETE
 LDX #BADARG
 JMP ERRMSG TELL THEM AND LEAVE
SETECM  INC     DONFLG
        BRA     OCONT
SYNERR  LDX     #SYNMSG
        JMP     ERRMSG
ENDCMD  INC     DONFLG  SET DONE FLAG
FORMAT LDA DRVNUM
 ORA #'0 MAKE ASCII
 STA DMSG PUT IN MSG
        LDX     #FMTMSG
        JSR     PSTRNG
        LDX     #STBLCK
        JSR     OUTHEX
        LDA     #'-
        JSR     PUTCHR
        LDX     #ENDBLK
        JSR     OUTHEX
 LDX #DRVMSG TELL THEM
 JSR PDATA
```

```
 LDY #0 INIT FREE SEC CTR
        LDA STBLCK GET SEGMENT START ADDR
        STA     GOODTK  LAST 'GOOD' TRACK #
TRKLP LDX #WINDOW SET BASE ACCESS ADDRESS
 STA DAT SET FOR TESTING
 LDB #$A5 GET TEST BYTE
        STB     $7FF,X PUT IT
        CMPB    $7FF,X IS IT SAME?
        BNE     NXTTRK NO, TRY NEXT TRACK
        STB     $FFF,X  TRY DIFFERENT ADDR JUST TO MAKE SURE
        CMPB    $FFF,X

        BNE     NXTTRK
 LDB #2 YES, FORMAT IT
        STA     GOODTK  THIS IS GOOD TRACK
SECLP STD ,X++ SET FWD LINK
 PSHS B SAVE
 LDB #256 COUNTER
CLRSEC CLR ,X+
 DECB CNT DOWN
 BNE CLRSEC
 PULS B GET BACK
 INCB BUMP
 LEAY 1,Y COUNT SECTORS
 CMPX #WINDOW+$1000 LAST ON TRACK?
 BLO SECLP NOT LAST
NXTTRK  LDB     GOODTK
        STB     MAXTRK  SAVE HIGHEST TRACK
        STB     DAT     PAGE IT IN
        LDB     #1      FIRST SECTOR, NEXT TRACK
 INCA BUMP TO NEXT TRACK
 BEQ MAXEND END OF ALL MEMORY
        CMPA    ENDBLK  DONE?
        BHI     MRKEND
SET STD WINDOW+$F00 SET LINK
 BRA TRKLP DO NEXT SEGMENT
MRKEND  LDA     GOODTK
        CMPA    STBLCK  MAKE SURE SOME GOOD MEMORY WAS FOUND
        LBEQ    NOMEM
 CLRA
 CLRB
 STD WINDOW+$F00

* MARK DIRECTORY END - ASSIGN ONLY ONE DIRECTORY SECTOR & LET
* FLEX EXTEND IT AS NECESSARY

        LDA     STBLCK  GET FIRST BLOCK
 STA DAT SET
 CLRA
 CLRB
 STD WINDOW+$100 MARK END OF DIR

* BUILD SIR

        LDA     STBLCK  GET FIRST TRACK
 STA DAT
 LDX #WINDOW
 CLR ,X+
 CLR ,X+ CLEAR OUT LINK
 LEAX 14,X
 LDU #VOLNAM SET VOLUME NAME
NAMLP LDB ,U+
 STB ,X+
 CMPU #VOLNAM+11
 BLO NAMLP
 LDD #1 SET VOLUME NUMBER
 STD ,X++
        LDA STBLCK SET FIRST AVAIL SECTOR
        LDB     #3
 STD ,X++
 LDA MAXTRK GET LAST TRK
 LDB #$10 MAX SEC ON TRK
STD ,X++
PSHS D SAVE
LEAY -2,Y REDUCE SEC CNT FOR SIR & DIR SEC
STY ,X++ SET SECTOR COUNT
LDD DATE SET DATE
STD ,X++
LDA DATE+2
STA ,X+
PULS D GET MAX TRK/SEC
```

```
 STD ,X++ SET MAX TRK/SEC
 BRA INIT

* PRESERVE EXISTING INFO

NOFMT   LDA     STBLCK  SET DAT FOR SIR
 STA DAT
 LDX #WINDOW+16 SET FOR COMPARE
 LDU #VOLNAM
CMPLP LDB ,U+
 CMPB ,X+
 BNE NOMTCH FAIL
 CMPU #VOLNAM+11
 BNE CMPLP NOT END, KEEP ON
 LDY 6,X GET SECTOR CNT
 LDA 11,X GET MAX TRK
 STA MAXTRK SET

* INITIALIZE SWITCHES

INIT LDA #STDDAT RESET DAT TO NORMAL
 STA DAT
        LDD     DSKVEC+1 ARE WE ALREADY 'IN'?
        CMPD    #READ
        BEQ     NOCLR   YES, DON'T CLEAR STARTING BLOCK TABLE
        LDU     #STBLKS
CLRSLP  CLR     ,U+
        CMPU    #STBLKS+10
        BLO     CLRSLP
NOCLR   LDA     #$FF
 STA SELECT INIT SWITCH TO NOT SEL
        LDU     #STBLKS PUT STARTING BLOCK IN TABLE
        LDB     DRVNUM
        LDA     STBLCK
        STA     B,U
 LDD #$1001 SET INIT TRK/SEC
 STD TRACK

* TELL THEM WE ARE DONE

 LDX #MSG
 JSR PSTRNG
 PSHS Y SAVE TOTAL SECS
 TFR S,X GET PTR TO SAVED
 CLRB SET ZERO FLG
 JSR OUTDEC OUT CNT

* MOVE OLD DISK I/O VECTORS AND REPLACE

 LDD DSKVEC+1 GET READ ADDR
 CMPD #READ IT IS US
 BEQ EXIT DONT MOVE VECTORS TWICE
 LDX #NEWVEC MOVE IN NEW VECTORS
MVNLP LDA ,X+
 LDB DSKVEC-NEWVEC-1,X SAVE OLD
 STA DSKVEC-NEWVEC-1,X SET NEW
 STB DSKVEC-NEWVEC+SIZE-1,X SET MOVED OLD
 CMPX #NEWVEC+SIZE DONE?
 BLO MVNLP NO
 BRA EXIT

NOMTCH LDX #NOTFM
 BRA ERRMSG
NOMEM LDX #NOMEMM
ERRMSG LDA #STDDAT SET DAT BACK TO NORMAL
 STA DAT
 JSR PSTRNG
     JMP WARMS ALL DONE INITIALIZING
EXIT    TST     DONFLG  ANYTHING ELSE ON COMMAND LINE?
        LBEQ    PARSE4  YES, CONTINUE PARSE
        JMP     WARMS   ELSE QUIT

* REMOVE I/O VECTOR CHANGES

REMOVE LDD DSKVEC+1 GET ADDR OF READ JMP
 CMPD #READ IS IT US?
 BNE EXIT NO, NOTHING TO DO
 LDX #DSKVEC INIT PTR
RMLP LDA SIZE,X
 STA ,X+ MOVE
 CMPX #DSKVEC+SIZE DONE?
```

```
 BNE RMLP NO
 BRA EXIT YES

DELETE  LDA     DRVNUM
        LDU     #STBLKS
        CLR     A,U
        LDA     #$FF    DESELECT
        STA     SELECT
        BRA     EXIT

* HEX INPUT SUBROUTINE - ONLY ACCEPTS FIRST TWO DIGITS

GETHEX  CLRB
        PSHS    B       TEMPORARY STORAGE
GHLOOP  JSR     NXTCH   GET CHARACTER
        BSR     GETDIG  GET DIGIT
        BCS     GHEXIT  EXIT IF NOT HEX
        INCB            BUMP COUNTER
        CMPB    #2
        BHI     GHLOOP  IGNORE DIGITS AFTER SECOND
        ASL     0,S     ALIGN MS NIBBLE
        ASL     0,S
        ASL     0,S
        ASL     0,S
        ORA     0,S     MERGE IN LS NIBBLE
        STA     0,S     SAVE IT
        BRA     GHLOOP

GHEXIT  PULS    A,PC    GET BYTE AND RETURN

GETDIG  CMPA    #'0
        BLO     NOTHEX
        CMPA    #'F
        BHI     NOTHEX
        CMPA    #'9
        BLS     SKPADJ
        CMPA    #'A
        BLO     NOTHEX
        SUBA    #7
SKPADJ  ANDA    #$F
        CLC
        RTS
NOTHEX  STA     TERMCH  STORE TERMINATING CHAR
        SEC
        RTS

PDLOOP  JSR     PUTCHR
PDATA   LDA     ,X+
        CMPA    #EOL
        BNE     PDLOOP
        RTS

DRVNUM  RMB     1       DRIVE # CURRENTLY BEING FORMATTED
MAXTRK  RMB     1       MAX TRACKS SIMULATED
STBLCK  RMB     1       STARTING BLOCK OF CURRENT 'DRIVE'
ENDBLK  RMB     1       ENDING    "    "    "    "
TERMCH  RMB     1       GETHEX TERMINATING CHARACTER
STFLAG  RMB     1       START BLOCK PROCESSED FLAG
DONFLG  RMB     1       COMMAND COMPLETELY PARSED FLAG
GOODTK  RMB     1       LAST GOOD TRACK

* MESSAGES

VOLNAM FCC 'HYPERDSKVOL'
FMTMSG FCC 'Formatting ',EOL
DRVMSG FCC ' as drive '
DMSG FCC '0',EOL
MSG FCC 'Total Sectors Available: ',EOL
NOMEMM FCC 'NO MEMORY AVAILABLE',EOL
BADARG FCC 'BAD OPTION',EOL
NOTFM FCC 'MEMORY NOT FORMATTED',EOL
SYNMSG  FCC     'SYNTAX ERROR - PLEASE RETYPE COMMAND',EOL

* NEW DISK DRIVER ROUTINE VECTORS

NEWVEC JMP READ
 JMP WRITE
 JMP VERIFY
 JMP RESTOR
 JMP DRIVE
 JMP CHKRDY
 JMP CHKRDY (QUICK)
 LBRA *+SIZE INIT, PASS THRU
 LBRA *+SIZE WARM, PASS THRU
 JMP SEEK
SIZE EQU *-NEWVEC TABLE SIZE
 PAG
* DISK DRIVER SIMULATOR FOR HYPERDISK

 ORG DSKVEC+SIZE LEAVE SPACE FOR ORIGINAL VECTORS
RD RMB 3 SPACE FOR MOVED VECTORS
WR RMB 3
VFY RMB 3
RES RMB 3
DRV RMB 3
CHK RMB 3
QIK RMB 3
 RMB 6 PASSED THRU
SK RMB 3

STBLKS  RMB     10       STARTING BLOCKS FOR DRIVES 0-9
SELECT RMB 1 DRIVE WHICH IS SELECTED, $FF IF NONE
TRACK RMB 1 SIMULATED TRACK REG, SET BY SEEK & RESTORE
SECTOR RMB 1 SIMULATED SECTOR REG
MAXSEC FCB $10 MAX SECTOR SIZE

CHKNUM  PSHS    B,U
        LDB 3,X GET DRIVE NUM
        LDU     #STBLKS
        LDA     B,U
        PULS    B,U,PC

CHKRDY  PSHS    A
        BSR     CHKNUM
        PULS    A
 BEQ CHK
 CLRB SET AS RDY
 RTS

DRIVE   PSHS    A
        BSR     CHKNUM
        PULS    A
        BNE SEL
        LDB     #$FF
        STB SELECT TURN US OFF
 BRA DRV GO DO REAL
SEL LDB 3,X TURN US ON
 STB SELECT
 CLRB SET NO ERROR
 RTS

RESTOR  PSHS    A
        BSR     CHKNUM
        BNE     RES1
        PULS    A
        BRA     RES     PASS THROUGH
RES1    LDD #$01 SET AT FIRST SECTOR
 STD TRACK
        PULS    A
 CLRB SET NO ERROR
 RTS

SEEK    PSHS    A
        LDA     SELECT  ARE WE SELECTED?
        CMPA    #$FF
        PULS    A
 BEQ SK NO, PASS IT ON
        PSHS    U
       LDU     #STBLKS GET BASE FOR STARTING BLOCKS
CMPD #SIRSEC IS IT SIR?
BNE SK1 NO
       LDA     SELECT  GET DRIVE #
       LDA     A,U     GET STARTING BLOCK
       LDB     #1      SET TO FIRST SEC
SK1 CMPD #DIRSEC IS IT DIRECTORY START?
 BNE SK2 NO
       LDA     SELECT
       LDA     A,U
       LDB     #2      SET TO SECOND SECTOR
SK2    PSHS    B
       LDB     SELECT
        CMPA    B,U
        PULS    B,U
 BLO NOTFND
        CMPB    MAXSEC
        BHI     NOTFND
 STD TRACK SET FOR R/W
 CLRB SET NO ERROR
 RTS
NOTFND LDB #$10 SET RNF ERROR
 RTS

VERIFY  PSHS    A
        LDA     SELECT
        CMPA    #$FF
        PULS    A
 LBEQ VFY NO, PASS ON
 CLRB YES, WAS GOOD
 RTS

WRITE   PSHS    A
        LDA     SELECT
        CMPA    #$FF
        PULS    A
 LBEQ WR NO
 BSR SEEK GO TEST AND SET
 BNE ERROUT
 PSHS U SAVE
 BSR RWSET
 EXG X,U SET PROPER TO-FROM REGS
 BRA XFER
READ    PSHS    A
        LDA     SELECT
        CMPA    #$FF
        PULS    A
 LBEQ RD NO, PASS ON
 BSR SEEK SEEK AND LIMIT TEST
 BNE ERROUT
 PSHS U SAVE
 BSR RWSET GO SET UP FOR OP
XFER TFR X,D GET TO ADDR
 INCA BUMP ONE SECTOR
 PSHS D SAVE FOR COMPARE
LOOP LDD ,U++ GET WORD
 STD ,X++ SET WORD
 CMPX ,S ARE WE DONE?
 BNE LOOP NO
 LEAS 2,S REMOVE COMPARE
 LDA #STDDAT
 STA DAT SET BACK NORMAL
 PULS U RESTORE
 CLRB SET GOOD
ERROUT RTS

RWSET LDA TRACK
 STA DAT SET DAT AT TRACK
 LDU #WINDOW SET SECTOR ADDR
 LDA SECTOR
 DECA BUMP BACK
 CLRB
 LEAU D,U CALC SEC START
 RTS

 END START
```

CLASSIFIED ADVERTISING

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New cost $1295.00 - ONLY **$759.00** ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

The following SWTP-6809 Flex/Uniflex components: 2-S/09+ mainframes, 256K board, 10meg Mini-Wini, X-12, 2-8212's, 3-64K boards, dual 8-inch diskette drive, ct-82, QUME 45cps printer with tractor, sheetfeeder, and MP-QP interface board.
Make offers to Richard Davidson, (517)332-5989.

***

SWT-S/09, 2MC, 128kRam, 2 x 8" dsdd and/or 2 x 5" sssd floppies, ready to use. Make me an offer.
Peter Keller Switzerland (Tel:01-984 29 84) (Tx:59887)

***

MEX6801 Support (development) system for Exorciser or Exorterm. Consists of Intercept, Control and Buffer modules, software, documentation. User System Evaluator (USE) capability, real time emulation, EXORbus compatible. List price MEX6801 $2700. For sale at $1200. Also 10 card slot power supply, rack mount chassis (M68MMLC) for $300.
Contact Karl Ritzinger (603)-434-2300 (NH) days.

***

DOUBLE DATE ISSUE

As some of you might have noticed, this issue carries two dates AUGUST/SEPTEMBER 1984. The reason being that we are growing on the newsracks. Because of date/time considerations we needed a little more time than our present issue dating allowed on the bookrack shelves. Hence, the double dating.

This will in no way affect your subscription frequency. You will still receive the proper number of issues. It will just be that if your subscription expired in say October '84 before, it will now expire in November '84. Nothing lost and a lot more exposure for our advertisers, and we sell a few more copies.

Hope you understand, nothing really changes as far as your subscription is concerned, it is just one of those technical things that occur as we 'grow'.

DMW

- - -

VDATE

FLEX Utility Reports File Creation Date

By Geoffrey A. Gass
5240 S. W. Dosch Rd.
Portland, Oregon 97201

You may not be aware of it, but the FLEX* operating system carefully notes down in the disk directory the creation date of every file you save. In the standard set of utilities furnished with FLEX 1.Ø (for the DMAF-1 8" disk), there is no means of accessing this information, except by single-stepping the directory and snooping in the $A89Ø-A97F region of the file sector buffer.

There is a DATE transient command, however, which allows you to read or alter FLEX' MO-DA-YR registers (this, by the way, is not the routine used when the system is first booted up: that's an input-only routine in the resident part of FLEX). As your command string is parsed by the DATE program, it jumps to an input routine if it sees something beyond the DATE or DATE.CMD filename, but simply reports the current date if there is no operand in the line.

It's a relatively simple operation for the program to look a little closer at the operand, and if it looks like a filename (starts with a letter or with one digit and a period), to use that name to open a file and extract the date information from the directory, processing it as it would the MO-DA-YR register.

*TM, Technical Systems Consultants, Inc.

There are a couple of traps to watch for. The TSC-furnished .SYS, .LOW and .CMD files have Ø-Ø-Ø as creation dates. The month-lookup routine can't handle "Ø", so a "Ø" date must be traped out.

And, if you have been playing around and using these directory bytes for something else, you'll also need to trap out any "month" value higher than $C, or the month routine will print out garbage -- maybe lots of it!

In the FLEX directory, the byte just before the month digit is not (yet) used by TSC. It's a handy place to keep track of revision numbers -- particularly on files which may be revised or updated several times in a day. If you train your assembler or text editor to extract this byte on every delete operation and plug in the incremented value when you open the file for writing, you will be able to check on this revision number (1 to 255) of any file, without having to download the file itself, just by calling VDATE,FILENAME.EXT.

If the revision byte is nonzero, VDATE will report:

Rev 35 April 22, 1981

— or whatever. The "Rev" title and number are omitted if the data byte is Ø.

The original DATE routine in loader format just barely slops over into a second sector — so there's plenty of room if you want to substitute this program for the original DATE.CMD, even on a full disk.

The VDATE routine here is coded and "ORG'd" for the SWTPC 68ØØ/DMAF-1. It is no doubt adaptable to the SWTPC mini-floppy and 68Ø9 versions, and perhaps to SSB FLEX as well -- but I haven't seen the code on these versions, and won't promise a thing.

But if you have the original target equipment, give it a try: you'll like it!

```
ØØØ1Ø                  NAM    VDATE      Modified DATE Utility
ØØØ2Ø              *          Rev Ø.3    March 28, 198Ø
ØØØ3Ø                  OPT    O,NOG

ØØØ5Ø              *Provides reporting of creation/revision date
ØØØ6Ø              *of disk files; and of revision numbers if
ØØØ7Ø              *implemented, as well as current date I/O.

ØØØ9Ø              *By Geoffrey A. Gass     Portland, Oregon 972Ø1

ØØ11Ø              *External References

ØØ13Ø       AC11   LTERM  EQU     $AC11     Last terminator or delimiter
ØØ14Ø       ACØ2   EOLCH  EQU     $ACØ2     EOL character
ØØ15Ø       ADØ3   WARMS  EQU     $ADØ3     FLEX re-entry
ØØ16Ø       A84Ø   FCB    EQU     $A84Ø     System FCB
ØØ17Ø       AD3F   RPTERR EQU     $AD3F     Error reporting routine
ØØ18Ø       AD48   INDEC  EQU     $AD48     Decimal input, binary conv.
ØØ19Ø       AD24   PCRLF  EQU     $AD24     Output CRLF
ØØ2ØØ       AD18   PUTCHR EQU     $AD18     Output character
ØØ21Ø       AD39   OUTDEC EQU     $AD39     Binary to decimal & out.
ØØ22Ø       ACØE   FMO    EQU     $ACØE     FLEX Date register
ØØ23Ø       ACØF   FDA    EQU     $ACØF
ØØ24Ø       AC1Ø   FYR    EQU     $AC1Ø
ØØ25Ø       AC14   LBUFP  EQU     $AC14     Line buffer pointer
ØØ26Ø       AD2D   GETFIL EQU     $AD2D     Pick up file spec
ØØ27Ø       AD33   SETEXT EQU     $AD33     Set default ext if none.
ØØ28Ø       B4Ø6   FMS    EQU     $B4Ø6     File management system
ØØ29Ø       AD36   ADDBX  EQU     $AD36     XR = XR + ACC B

ØØ31Ø A1ØØ                ORG     $A1ØØ

ØØ33Ø A1ØØ 2Ø Ø7   VDATE  BRA     DATE1     Skip registers

ØØ35Ø A1Ø2 Ø1      V      FCB     1         Version
ØØ36Ø A1Ø3 ØØ      REV    FCB     Ø         Rev Ø for files so flagged
ØØ37Ø A1Ø4 ØØ      MO     FCB     Ø         From FLEX or file
ØØ38Ø A1Ø5 ØØ      DA     FCB     Ø
ØØ39Ø A1Ø6 ØØ      YR     FCB     Ø
ØØ4ØØ A1Ø7 ØØ      NUMH   FCB     Ø         Working register
ØØ41Ø A1Ø8 ØØ      NUML   FCB     Ø         LSB

ØØ43Ø A1Ø9 B6 AC11 DATE1  LDA A   LTERM     Get last delimiter
ØØ44Ø A1ØC 81 ØD          CMP A   #$ØD      C/R?
ØØ45Ø A1ØE 27 5A          BEQ     DATO      If so, just report date.
ØØ46Ø A11Ø 81 ACØ2        CMP A   EOLCH     EOL character?
ØØ47Ø A113 27 55          BEQ     DATO      No input: output only
ØØ48Ø A115 FE AC14        LDX     LBUFP     Buffer pointer
ØØ49Ø A11D A6 ØØ          LDA A   Ø,X       Have a look
ØØ5ØØ A11A 81 4Ø          CMP A   #$4Ø      Alpha?

ØØ51Ø A11C 22 Ø6          BHI     FILOP     Probably a filename
ØØ52Ø A11E A6 Ø1          LDA A   1,X       Look at next-following
ØØ53Ø A12Ø 81 2E          CMP A   #$2E      Period?
ØØ54Ø A122 26 Ø3          BNE     DATE2     Probably date input
ØØ55Ø A124 7E A1CF FILOP  JMP     FILDAT    Must be filename.
```

```
ØØ57Ø A127 8D 2F   DATE2  BSR    NUMIN    Get number from buffer
ØØ58Ø A129 25 2Ø          BCS    SYNERR   C set: bad input
ØØ59Ø A12B 81 ØC          CMP A  #$ØC     Over 12?
ØØ6ØØ A12D 22 1C          BHI    SYNERR   Does not compute
ØØ61Ø A12F B7 ACØE        STA A  FMO      Into FLEX date register
ØØ62Ø A132 8D 24          BSR    NUMIN    Get next digit(s)
ØØ63Ø A134 25 15          BCS    SYNERR   Bummer?
ØØ64Ø A136 81 1F          CMP A  #$1F     Over 31?
ØØ65Ø A138 22 11          BHI    SYNERR   Don't dig long months
ØØ66Ø A13A B7 ACØF        STA A  FDA      February 31 is OK.
ØØ67Ø A13D 8D 19          BSR    NUMIN    For year
ØØ68Ø A13F 25 ØA          BCS    SYNERR   C flags NG or no data
ØØ69Ø A141 81 63          CMP A  #$63     99 in hex
ØØ7ØØ A143 22 Ø6          BHI    SYNERR   Actual date + 256 is OK.
ØØ71Ø A145 B7 AC1Ø        STA A  FYR      To FLEX
ØØ72Ø A148 7E ADØ3        JMP    WARMS    All done.

ØØ74Ø A14B CE A84Ø SYNERR LDX    #FCB     System control block
ØØ75Ø A14E C6 1A          LDA B  #$1A     Syntax error (#26)
ØØ76Ø A15Ø E7 Ø1   SYNER1 STA B  1,X      Into Error Status byte
ØØ77Ø A152 BD AD3F        JSR    RPTERR   FLEX routine
ØØ78Ø A155 7E ADØ3        JMP    WARMS    Back to FLEX.

ØØ8ØØ A158 BD AD48 NUMIN  JSR    INDEC    Input & binary conversion
ØØ81Ø A15B 25 ØC          BCS    NUMEX    Bad input
ØØ82Ø A15D 5D             TST B           Check for no-input
ØØ83Ø A15E 27 Ø8          BEQ    NREJ     Delimiter -- no digits
ØØ84Ø A16Ø FF A1Ø7        STX    NUMH     Four hex digits
ØØ85Ø A163 B6 A1Ø8        LDA A  NUML     Get LSB only
ØØ86Ø A166 ØC             CLC             All's well
ØØ87Ø A167 39             RTS
ØØ88Ø A168 ØD      NREJ   SEC             Bounce flag
ØØ89Ø A169 39      NUMEX  RTS

ØØ91Ø              *DATO will output REV (if any), MO, DA, YR.

ØØ93Ø A16A CE ACØE DATO   LDX    #FMO     Start of FLEX date registers
ØØ94Ø A16D A6 ØØ   DATO1  LDA A  Ø,X      Entry for creation date
ØØ95Ø A16F B7 A1Ø4        STA A  MO       Local register
ØØ96Ø A172 EE Ø1          LDX    1,X      Get DA, YR
ØØ97Ø A174 FF A1Ø5        STX    DA
ØØ98Ø A177 BD AD24        JSR    PCRLF    Do a CRLF
ØØ99Ø A17A F6 A1Ø3        LDA B  REV      Look at REV register
Ø1ØØØ A17D 27 ØE          BEQ    DATO2    Nothing? Omit mention.

Ø1Ø1Ø A17F CE A2Ø4        LDX    #REVT    'Rev '
Ø1Ø2Ø A182 BD A1BD        JSR    STRNGO   Output text
Ø1Ø3Ø A185 17             TBA             Swap # into A
Ø1Ø4Ø A186 8D 3A          BSR    NUMO     Output it
Ø1Ø5Ø A188 86 2Ø          LDA A  #$2Ø     Space
Ø1Ø6Ø A18A BD AD18        JSR    PUTCHR   Output that

Ø1Ø8Ø A18D B6 A1Ø4 DATO2  LDA A  MO       Get month (binary)
Ø1Ø9Ø A19Ø 27 22          BEQ    ENDPRT   ØØ? Print one zero.
Ø11ØØ A192 CE A2ØE        LDX    #MOTAB   Table of names
Ø111Ø A195 4A      NEXN   DEC A           Count 'em out
Ø112Ø A196 27 Ø8          BEQ    DPRINT   Found it? Print it.
Ø113Ø A198 Ø8      LOOP   INX             Step to next
Ø114Ø A199 6D ØØ          TST    Ø,X      Looking for ØØ delimiter
Ø115Ø A19B 26 FB          BNE    LOOP
Ø116Ø A19D Ø8             INX             Past delimiter
Ø117Ø A19E 2Ø F5          BRA    NEXN     Try this one

Ø119Ø A1AØ 8D 1B   DPRINT BSR    STRNGO   Run month name
Ø12ØØ A1A2 86 2Ø          LDA A  #$2Ø     Space
Ø121Ø A1A4 BD AD18        JSR    PUTCHR   Output it
Ø122Ø A1A7 B6 A1Ø5        LDA A  DA
Ø123Ø A1AA 8D 16          BSR    NUMO
Ø124Ø A1AC CE A2Ø9        LDX    #CENTUS  Text string ', 19'
Ø125Ø A1AF 8D ØC          BSR    STRNGO   Run it
Ø126Ø A1B1 B6 A1Ø6        LDA A  YR
Ø127Ø A1B4 8D ØC   ENDPRT BSR    NUMO
Ø128Ø A1B6 7E ADØ3        JMP    WARMS    All done.

Ø13ØØ A1B9 BD AD18 STRNG1 JSR    PUTCHR
Ø131Ø A1BC Ø8             INX
Ø132Ø A1BD A6 ØØ   STRNGO LDA A  Ø,X      Entry point
Ø133Ø A1BF 26 F8          BNE    STRNG1
Ø134Ø A1C1 39             RTS             ØØ is delimiter

Ø136Ø A1C2 5F      NUMO   CLR B           No leading zeros
Ø137Ø A1C3 F7 A1Ø7        STA B  NUMH     No garbage in MSB
Ø138Ø A1C6 B7 A1Ø8        STA A  NUML     Data
Ø139Ø A1C9 CE A1Ø7        LDX    #NUMH    Pointer for 4 digits
Ø14ØØ A1CC 7E AD39        JMP    OUTDEC   Print the last 1 or 2

Ø142Ø              *FILDAT looks up a file creation date

Ø144Ø A1CF CE A84Ø FILDAT LDX    #FCB     Use system FCB
Ø145Ø A1D2 BD AD2D        JSR    GETFIL   Get filename from buffer
Ø146Ø A1D5 24 Ø4          BCC    FILO1    OK? Set extension
Ø147Ø A1D7 C6 15          LDA B  #$15     Code for filename error
Ø148Ø A1D9 2Ø 26          BRA    FERR
Ø149Ø A1DB 86 Ø1   FILO1  LDA A  #1       Open for read
Ø15ØØ A1DD A7 ØØ          STA A  Ø,X      Into FCB
Ø151Ø A1DF 86 ØØ          LDA A  #Ø       Default extension (.BIN)
Ø152Ø A1E1 BD AD33        JSR    SETEXT   Set default if no ext.
Ø153Ø A1E4 BD B4Ø6        JSR    FMS      Open file
Ø154Ø A1E7 26 16          BNE    FERR1
Ø155Ø A1E9 86 Ø4          LDA A  #4       Close-file code
Ø156Ø A1EB A7 ØØ          STA A  Ø,X
Ø157Ø A1ED BD B4Ø6        JSR    FMS      Do it
B158Ø A1FØ 26 ØD          BNE    FERR1    Problems?
Ø159Ø A1F2 A6 18          LDA A  $18,X    Byte 24, non-FLEX Rev #
Ø16ØØ A1F4 B7 A1Ø3        STA A  REV      FLEX files will have ØØ
Ø161Ø A1F7 C6 19          LDA B  #$19
Ø162Ø A1F9 BD AD36        JSR    ADDBX    Point to creation date
Ø163Ø A1FC 7E A16D        JMP    DATO1    Load MO-DA-YR and output.

Ø165Ø A1FF E6 Ø1   FERR1  LDA B  1,X      Get error code
Ø166Ø A2Ø1 7E A15Ø FERR   JMP    SYNER1   Report & exit

Ø168Ø A2Ø4 52      REVT   FCC    /Rev /
Ø169Ø A2Ø8 ØØ             FCB    Ø
Ø17ØØ A2Ø9 2C      CENTUS FCC    /, 19/
Ø171Ø A ØD ØØ             FCB    Ø

Ø173Ø A2ØE 4A      MOTAB  FCC    /January/
Ø174Ø A215 ØØ             FCB    Ø
Ø175Ø A216 46             FCC    /February/
Ø176Ø A21E ØØ             FCB    Ø
Ø177Ø A21F 4D             FCC    /March/
Ø178Ø A224 ØØ             FCB    Ø
Ø179Ø A225 41             FCC    /April/
Ø18ØØ A22A ØØ             FCB    Ø
Ø181Ø A22B 4D             FCC    /May/
Ø182Ø A22E ØØ             FCB    Ø
Ø183Ø A22F 4A             FCC    /June/
Ø184Ø A233 ØØ             FCB    Ø
Ø185Ø A234 4A             FCC    /July/
Ø186Ø A238 ØØ             FCB    Ø
Ø187Ø A239 41             FCC    /August/
Ø188Ø A23F ØØ             FCB    Ø
Ø189Ø A24Ø 53             FCC    /September/
Ø19ØØ A249 ØØ             FCB    Ø
Ø191Ø A24A 4F             FCC    /October/
Ø192Ø A251 ØØ             FCB    Ø
Ø193Ø A252 4E             FCC    /November/
Ø194Ø A25A ØØ             FCB    Ø
Ø195Ø A25B 44             FCC    /December/
Ø196Ø A263 ØØ             FCB    Ø,$FF

Ø198Ø AØ48                ORG    $AØ48
Ø199Ø AØ48 A1ØØ           FDB    $A1ØØ       Transfer address.
Ø2ØØØ                     END

TOTAL ERRORS ØØØØØ
```

Dear Editor,

At one time I was very interested in the 6809 CPU. I owned a computer from SWTPC which included dual disk drives and the FLEX operating system. Needing a better version of a 6809 based computer I bought a Radio Shack 64k Color Computer. At the time I had received a few free issues of the Color Micro Journal and read about the deficiencies in the Radio Shack disk system. I decided to buy their new disk controller and mate it to my dual floppy. The dealer seemed to think that was O.K., and that was when the trouble started. He did not realise that Radio Shack keeps things to themselves. After going to a lot of trouble to get all the parts he found that a manual was not available. He was a decent guy, but I still got so mad I set the entire computer aside and have not touched it since. After 25 years as an Electronic Engineer I have learned that some things are not worth bothering with. Radio Shack could have done a lot with that computer, but seems to be just too cheap and greedy.

This brings me to the substance of my letter. Why does "American Management" make such bad decisions? What happened to personal pride in a job well done? My experience has been that even though engineering is sometimes difficult, it is infinitely more difficult to talk to management. Witness the millions that have been lost in the computer field by c panies that "had it made". I have seen people who seem much smarter than I am (and whom I personally liked) make absolutely unworkable decisions. No wonder the Japanese can penetrate our mar et so easily.

Of course we all make mistakes. When I do, I make no bones about it. Hiding an error hurts everyone, and can end up costing more than most people are willing to admit.

In the publication you work for you come in constant contact with software and hardware that just does not work right. There is no excuse for this, but it happens all the time. I have seen equipment costing more than $10,000 which was not even a good boat anchor. At my last job I saw a computer driven test system built that was a joke from the start. When all the smoke cleared the grapevine had it that a quarter million went down the tubes. I knew most of the people involved. Not one of them was unintelligent or dishonest, but still the fiasco went ahead.

Perhaps I should not unload all this on you. You are probably quite busy, and have heard all this before. What I am really trying to find out is the description of the new Radio Shack Color Computer disk controller, and whether the computer is worth developing hardware and software for. After working with FLEX, CPM, OS8, RTS8 and proprietary operating systems and building working hardware, I feel I have the qualifications needed.

Eds Note: This letter was sent to us with a request that this letter not be published since the writer felt that by having his name appear at the bottom, it might not be too difficult for someone to figure out which company he worked for and possibly the people he mentioned. Since some of the thoughts expressed in this letter are shared by many in the SS50 industry, I have taken the responsibility of inserting it into the Bit Bucket.

Sincerely,
Larry E. Williams
Exec. Editor

- - -

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

INTROL-C/6809,Version1.5

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident compilers are available under **Uniflex, Flex** and **OS9.**

Cross-compilers are available for **PDP-11/UNIX** and **IBM PC/PC DOS** hosts.

Trademarks:
Introl-C, Introl Corporation
Flex and Uniflex, Technical Systems Consultants
OS9, Microware Systems
PDP-11, Digital Equipment Corp.
UNIX, Bell Laboratories
IBM PC, International Business Machines

For further information, please call or write.

INTROL
CORPORATION

647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

OMEGASOFT INDUSTRIAL STRENGTH PASCAL FOR 6809/68000

OMEGASOFT PASCAL

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming more critical that a high level language be used. Find out why over 1000 companies have switched to OmegaSoft Pascal for their demanding applications.

WHY PASCAL?

Pascal was designed to teach students how to write structured programs that are easy to read and maintain. In the past decade it has also proved to provide the same advantages in industrial applications.

EXTENSIONS

OmegaSoft has taken the Pascal framework and expanded the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The byte data type allows you to directly address bytes in memory or I/O devices. The common arithmetic operations can be used for bytes along with shift left, shift right, "and", "or", "eor", and complement operators. These operators are also available for integer and hex (2 byte unsigned) numbers.

Longintegers are four byte signed numbers useful for extended range arithmetic commonly needed for machine control. Functions have been added to allow conversion between the various data types. Dynamic length strings allow complex text manipulation and allow effective interactive I/O.

Variables can be placed either on the data stack (default), at an absolute address in memory (for I/O), in base page, relative to the program counter (for constant tables), or defined in another module.

FEATURES

The compiler generates assembly language for assembly and linking to run on the target system. Since a true relocating assembler and linking loader is used, only those runtime modules required are automatically linked in, providing a much smaller object module than other compilers.

Large Pascal programs can be split up into conveniently sized modules to speed the development process. Procedures, functions, and variables can be referenced between Pascal modules and assembly language modules by using Pascal directives.

Full source code is included for the runtime library, the debugger, and other support utilities.

ISO COMPATIBILITY

OmegaSoft Pascal has been tested using the pascal Validation Suite. The Suite is a collection of over 400 Pascal programs designed to test the quality of Pascal Compilers and their runtime systems for compliance with the ISO (International Standards Organization) Pascal standard. OmegaSoft is the only supplier of 6809 native Pascal compilers that publishes this report in its instruction manual.

DEBUGGER

The compiler package includes an interactive, symbolic debugger. The debugger , allows setting of breakpoints, displaying and changing variables, and tracing statements. The debugger allows very fast turnaround for programs to be run on the host system.

6809 TARGET SYSTEM

The target system may be any 6809 system. No specific I/O devices are required. The output code is re-entrant and rom-able, perfect for single-board systems up to large development systems. There are no charges for use of the output of the compiler or the object of the runtime library in your products.

6809 HOST SYSTEM

The host system must be 6809 based and have at least 48K of ram (56K recommended) and run one of the following operating systems: MDOS, XDOS, OS—9, or FLEX. Priced from $425.

6809 SUPPORT PRODUCTS

The OmegaSoft Relocatable Assembler and Linking Loader is designed to support the Pascal Compiler Package and can also be used for general assembly language program development. Priced from $125.

OmegaSoft's Screen Editor supports smart terminals and comes complete with the Pascal source. Priced from $90.

For complex real-time applications, the Multi-tasking Kernel provides task scheduling, inter-task communications, and resource interlocking. The Kernel is a runtime library that is accessible as Pascal functions and procedures (with full source included). Priced from $175.

68000 SOFTWARE

A Cross Pascal package is available that runs on a 6809 host system and generates code for a 68000, 68008, or 68010. This package does not include a debugger, but does include a Relocatable Assembler and Linking Loader. Priced from $600.

A Resident 68000 Pascal package is available to run under VERSAdos, with support coming soon for OS-9/68000, CP/M-68K, and UNIX. This package will include the Compiler, Relocatable Assembler, Linking Loader, Debugger, and Screen Editor. Priced from $900.

Dealer and OEM inquiries invited. OmegaSoft products are also available from distributors in Australia and Western Europe, call or write for more information.

OmegaSoft products to run on Motorola development systems are available from Motorola systems distributors in Europe.

OMEGASOFT
CERTIFIED SOFTWARE CORP.
P.O. Box 842
Camarillo, CA 93010
Tel: (805) 987-6426
Telex: 467013

TM: OmegaSoft is a trademark of Certified Software Corporation. MDOS, XDOS, and VERSAdos are trademarks of Motorola. OS-9 and OS-9/68000 are trademarks of Microware. FLEX is a trademark of TSC. CP/M-68K is a trademark of DRI. UNIX is a trademark of A T & T.

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

ASSEMBLERS

Southeast Media

ASTRUK09

A "Structured Assembler for the 6809" which requires the TSC Macro Assembler. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apperent. Re. '68' Micro Journal, Sept. '83 (program was called "STASM09"; has been renamed due to conflicts).

A User reports

"... I'm very pleased and am now writing almost exclusively in (ASTRUK09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUK09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUK09) and the structure sure helps in understanding the logic!"

F, CCF - **$99.95**

TSC

Macro Assembler

The FLEX STANDARD Assembler. F,CCF $50.00

Relocating Assembler w/Linking Loader

Use with many of the C and Pascal Compilers. F,CCF $150.00

Great Plains Comp. Co.

RRMAC

Relocating, Recursive-Macro Assembler and Linking Loader.
F,CCF $120.00; w/Source $240.00

OmegaSoft

#RALL1

Relocating Assembler and Linking Loader
F,CCF $125.00: for One Year Maint., add $50.00

Windrush Micro Systems

MACE, by Graham Trott. F,CCF - **$98.00**

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGNAM (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer		SS-50 Bus (all w/ Source)
CCO (32K Req'd) Obj. Only	$49.00	F. $99.00
CCF, Obj. Only	$50.00	U. $100.00
CCF, w/Source	$99.00	O. $101.00
CCO, Obj. Only	$50.00	

ALL Computer Systems Consultants Software
runs on the Color FLEX Systems
ALL in stock
call 800-338-6800
for IMMEDIATE DELIVERY

Computer Systems Center

DYNAMITE +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Boundaries" during successive disassemblies; can use a Label File which automatically replaces a Hex location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF,	Obj.	Only	$100.00
CCO,	"	"	$59.95
F,	"	"	$100.00
O,	"	"	$150.00
U,	"	"	$300.00

COMPILERS & DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A combination Editor/Compiler/Debugger, all in **ONE PACKAGE**; provides a totally **INTERACTIVE** Program Development Cycle. The Single-Pass Compiler supports large Symbol Names; Variable Types; Pointers; Control Structures (similar to 'C' or 'Pascal'); Stack, A- ,B-, and D-Register manipulation; etc. The Source-Oriented Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool which provides for the maximum utilization of the power of the 6809.

F. CCF - **$198.00**

Whimsical Developments

WHIMSICAL

Need the Ease of Design and Maintainability of "Structured Programming" AND the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his review of WHIMSICAL in the Sept. '83 Issue of '68' Micro Journal that, except for the lack of floats, "..., **I have to give this one VERY high rating, ...".** It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.00

'C' Compilers

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System (lacking ONLY "bit-fields"), including an Assembler. Requires the TSC Relocating Assembler IF the user wishes to implement his own Libraries.

F and CCF - **$295.00**

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, "68" Micro Journal, May '83). Floats, etc.

F, CCF, and O - $375.00
U - $425.00
One Year Maint. - $[illegible]

VISA MasterCard

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends ——
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

PASCAL Compilers

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCP - $200.00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCP 5" - $190.00
F 8" - $265.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCP - $425.00
One Year Maint. - $100.00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

U - $219.95

UTILITIES

Southeast Media

Basic09 XRef

This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a Program List Utility which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires Basic09 or RunB for operation.

0 and CCO - Obj. Only -- $39.95
0 and CCO - w/ Source - $79.95

Southeast Media

OS-9 VDisk

Give your OS-9 Level I System the speed of memory access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - Level I ONLY.

O, obj. only - $79.95
w/ Source - $149.95

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ **from**, and WRITE **to**, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASIC09 (with Source), that performs the following functions;

REFORMAT: A BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

F and CCF - $79.95

Southeast Media

COPYMULT

--- Copy **LARGE** Disks to several **smaller** disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (8" or 5") $99.50

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal
Features;

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

VISA MasterCard

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

South East Media

5900 Cassandra Smith Rd.
Hixson, TN 37343
for Information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either caloriex and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Presant Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

Southeast Media

XDATA

A COMMUNICATION Package
for the UniFLEX Operating System

Allows UniFLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

-- Verifies Transmission integrity using checksum or CRC
-- Automatically Re-Transmits bad blocks
-- Transmits data in 128 byte blocks

U - $299.99

Southeast Media

- JUST
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

-Output the "Formatted" Text to the Display for format analysis and change.
-Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
-"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); providas for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
-Automatic compensation for a "Double Width" printed line.
-Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
-Use with **ANY** Editor.
-Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

Lucidata

PASCAL UTILITIES
Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F end CCF - $25.00

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of lar9e programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

Lucidata

COPYCAT
Pascal NOT required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directorias, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):
Copy a File with CRC Errors, so it can possibly be salvagad; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on tha Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order).

F and CCF - $50.00

WORD PROCESSORS

Alford and Associetes

SCREDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Fextures Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the Data-Comp and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what-you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00

F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.

F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.

F, CCF, O - $145.00 U - $195.00

VISA MasterCard

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
For Ordering

South East Media

5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Great Plains Computer Co.

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

F, CCF, O - $145.00
U - $195.00

Southeast Media

SPELLB "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELLB is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you KNOW it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F and CCF - $129.95

Great Plains Computer Co.

SPELL

Fast Computer Dictionary -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

F, CCF, OS/9 - $125.00
U - $175.00

DATA BASE MANAGEMENT SYSTEMS

Westchester Applied Business Systems

XDMS

Possibly one of the most powerful Database Management Systems available, this machine language program is small enough to operate on a **single sided 5" disk**, yet provides the **speed** of M.L. and power limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. **XDMS Level I** provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and Producing Reports. **XDMS Level II** adds the POWERFUL "GENERATE" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95
XDMS System Manual only - $24.95

Great Plains Computer Co.

INFOMAG DBMS

An XBASIC, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This Time Proven DBMS will become the "Work Horse" of your Software Stable.

F and CCF $295.00
U $395.00

ACCOUNTING PACKAGES

Great Plains Computer Co. and Universal Data Research, Inc. both have Business Packages written in TSC XBASIC for FLEX, CoCo FLEX, and UniFLEX ----

Computer Systems Consultants

BASIC UTILITY PROGRAMS

Ten BASIC Programs to:

A BASIC Resequencer with EXTRAs over "RENUM"; works with ALL Versions of FLEX BASIC AND the Precompiler, checks for missing label definitions, processes Disk to Disk instead of in Memory.

Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $25.00
UniFLEX - $50.00

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $150.00

The Virginia Company ***Bizpack***

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires TSC's Extended BASIC.

F and CCF - $135.00
with Source - $250.00

** SPECIAL **
Purchase XBASIC and BIZPACK together for $221.50
-- a Savings of $13.50 --

Availability Legends --
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

*** SPECIAL ***
Purchase XBASIC and BIZPACK together for $221.50
— a Savings of $13.50 --

Computer Systems Consultants
TABULA RASA SPREADSHEET
TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires **TSC's Extended BASIC**.

F and CCF - **$100.00**, U - **$125.00**

Computer Systems Center
DYNACALC
THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using **DYNACALC**). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is **FAST**. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.

F and SPECIAL CCF - $200.00
CoCo DOS - $99.95
O - $250.00
U - $237.00

ODDS & ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY
This package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen Input/Output capabilities of **TSC's Extended BASIC** programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - **$50.00**, U - **$75.00**

Computer Systems Consultants
FULL SCREEN MAILING LIST
The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires **TSC's Extended BASIC**.

F and CCF - **$100.00**, U - **$110.00**

COLOR COMPUTER SOFTWARE

Stearns Electronics
FORTH
Intrigued by Forth??? Here is a FORTH package **tailored** to the Color Computer! This package is supplied on Tape, with instructions for transferring it to **disk** if you wish. Written primarily in machine language, it's **speed is unparalleled**. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in learning FORTH, a **Trace Feature is provided which is invaluable.** If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - **$58.95**

Custom Software Engineering, Inc.
Color Computer **GRAPHIC SCREEN PRINT** Programs
Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
GSPRE for Epson w/ Graftrax and Graftrax +	$9.95
GSPRG for Gemini 10 and 15	$9.95
GSPRP for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.
DATE-O-BASE CALENDAR Program
A Menu Driven **EXTENDED BASIC** Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A **Graphic Calendar** shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95
DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

Custom Software Engineering, Inc.
That's INTEREST-ing
Interested in INTEREST (the Money Kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - **$29.95**

Custom Software Engineering, Inc.
DISK DATA HANDLER 64K
An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK - **$54.95**

Custom Software Engineering, Inc.
DISK DOUBLE ENTRY
DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DDE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include commente. Screen reports allow review of past transactions and current balances.

DISK - **$44.95**

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Computer Systems Center

DYNASHARE

— Multi-User, Multi-Tasking with FLEX —

Southeast Media is now shipping DYNASHARE FROM STOCK - the multi-user, multi-tasking capability of DYNASHARE allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new language or Operating System syntax. DYNASHARE, as its name implies, allows true "time-sharing" operation under the popular FLEX operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! DYNASHARE also provides some fringe benefits that will be greatly appreciated by FLEX users, including type-ahead, command line editing, and instant response to "escape".

DYNASHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user and/or task. Fact is, you still use FLEX just like you always have! DYNASHARE is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. What DYNASHARE does do is give FLEX users a **low-cost way to use existing software in a multi-user, multi-tasking environment**, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, DYNACALC spread-sheet package, and so on are still good.

NOTE -- The **initial release** of DYNASHARE is for SWTPC S/09 Computers, but versions will also be available for other popular extended-memory (up to 1024K) systems, such as HELIX and GIMIX. A minimum of 128K of RAM will be required with ALL versions. DYNASHARE requires **64K of RAM for each active task**: thus a 256K system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from **Southeast Media** - $200.00

AUTHORS - PROGRAMMERS
QUALITY SOFTWARE NEEDED
FLEX - UniFLEX - OS/9 - Color Computer

For the past several months, we at the **Southeast Media Division** of **Computer Publishing, Inc.** (CPI), the parent company of **'68' MICRO JOURNAL** and **COLOR MICRO JOURNAL**, have debated expanding our software distribution business. Many other magazines have been doing so for years (in fact, MOST were in the Software Distribution Business BEFORE they began to publish a Magazine). Presently there are many fine examples of software that has been developed by YOU, our readers, that will never see the "light of day" due to the **Cost of Advertising** and **TIME and Cost** involved in the **production, distribution,** and **Customer SUPPORT** of that software unless SOMEONE, with enough exposure and the willingness to continually advertise, runs with the ball.

Software is the "backbone" for the REAL utilization of any Computer System, and ours are no exception! This has been no simple decision. While we realize that there could be some conflict with some of our advertisers, we ALSO hear a LOUD and CONTINOUS cry for **HELP** from our Readers. From day one, the **foremost concern** of '68' MICRO JOURNAL has been it's **READERS!** Therefore, our **Southeast Media Division** will accept, for appraisal for possible Distribution, 6809 software; Games, Utilities, Software Development, Business Application Programs, etc.

In the past there has been **too much** software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in royalty payments, promptly paid, and in customer support for his product.

Of course, we will expect, no -- **DEMAND**, that the author keep the product free of errors (bugs), and maintain it in a prompt and business like manner. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a "single" sale product. In this event it will be combined with other like programs, and offered as a package. In that event, the royalties will be split between the various authors.

If you have software that you feel will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" — **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99% code already drifting around.

If your software is **READY** contact:
Bob Nay, Don Williams, or Tom Williams

Southeast Media is a division
of **Computer Publishing, Inc.** (CPI),
a family of 100% 68XX support facilities.

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

CSC

UniFLEX software prices include maintenance for the first year.

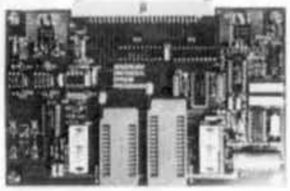
DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

Do it now!
U.S. MAIL

WINDRUSH MICRO SYSTEMS

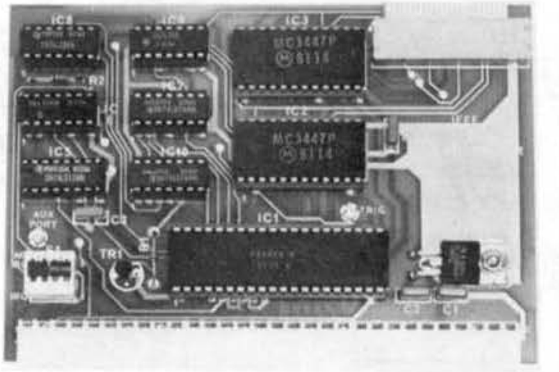
UNIVERSAL EPROM PROGRAMMER



- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 2732A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 27128.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair planar cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
  a. MOVE blocks of memory within the buffer.
  b. READ an EPROM into the buffer.
  c. VERIFY an EPROM against the buffer.
  d. EXAMINE and change the contents of the buffer.
  e. DUMP the contents of the buffer in HEX and ASCII.
  f. FILL a selected area of the buffer with a specified charecter.
- Software available for all versions of SSB DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customizing.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

IEEE-488



- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:

- Talker	- Serial Poll	- Single or Dual Primary Address
- Listener	- Parallel Poll	- Secondary Address
- System Controller	- Group Trigger	- Talk only...Listen only

- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9
EDITOR/COMPILER/DE-BUGGER

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is organized as a tutorial with plenty of examples.
- Fast single pass compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files
- Signed and unsigned BYTEs and INTEGERs, 32-bit floating point REALs.
- Vectors (single dimension arrays) and Pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/.XOR).
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (CCR) and (DREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library function modules are supplied: IOSUBS, BITIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column. Need we say more?

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly inter-active environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPROC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from microware:

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%
- Supports interleaved assembly language programs.
- The TSC relocating assembler/linking loader (SP09-17) is REQUIRED.

MACE	(6809 FLEX only)..	$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!)............	$198.00
'C'	(a 56K system and the TSC SP09-17 package is req'd)....	$295.00
IEEE-488	with IEEE-488 cable assembly............................	$298.00
SS-30	Universal EPROM programmer w/one version of software...	$375.00
EXORciser	Universal EPROM programmer w/one version of software...	$395.00
SOFTWARE	Drivers for a 2nd, 3rd, or 4th operating system........	$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, AND ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporations, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

An SS-50C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES,
NORTH WALSHAM, NORFOLK,
ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97360 SHARET G

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

New!

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

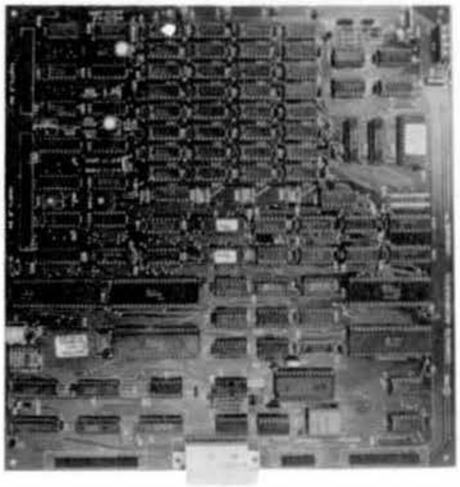
64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

BLANK PC BOARD

$99.95

WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.



$329.00

COMPLETE KIT! FULLY SOCKETED.

ALL OPTIONS ARE STANDARD. NO EXTRAS TO BUY!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful Instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power Is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- 64K RAM using 4116 RAMS.
- 6809E Motorola CPU.
- Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ASCII keyboard parallel input interface. (6522)
- Serial I/O (6551) for RS232C or 20 MA loop.
- Centronics compatible parallel printer interface. (6522)
- Buss expansion interface with DMA channel. (6844)
- Dual timer for real time clock application.
- Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

64K SS-50 STATIC RAM

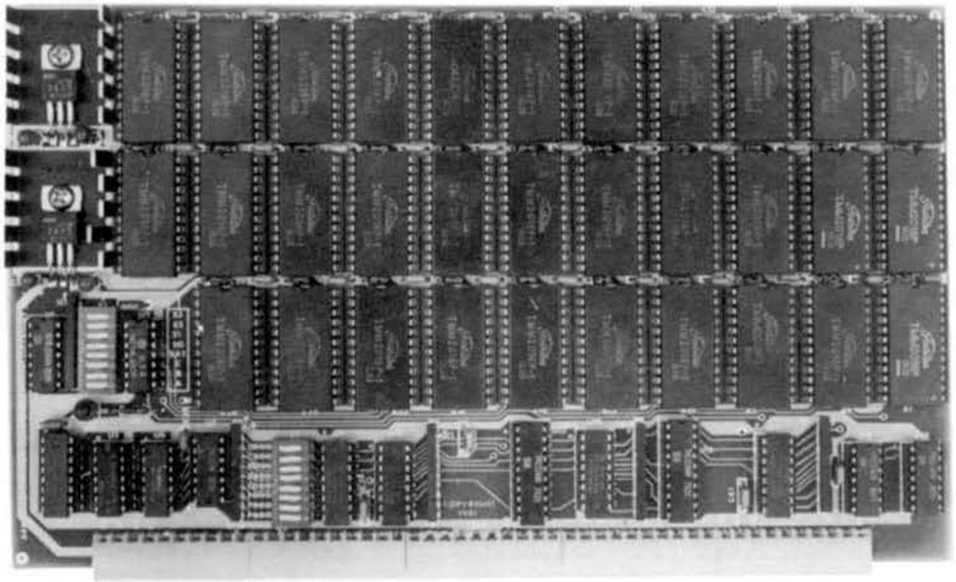
NEW!

$179.00
(48K KIT)

NEW!

LOW POWER!

RAM OR EPROM!



BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
★ Fully supports Extended Addressing.
★ 64K draws only approximately 500 MA.
★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
★ Board is configured as 3-16K blocks and 8-2K blocks (wi hin any 64K block) for maximum flexibility.
★ 2716 EPROMs may be installed anywhere on Board.
★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
★ One Board supports both RAM and EPROM.
★ RAM supports 2MHZ operation at no extra charge!
★ Board may be partially populated in 16K increments.

56K	$219
64K	$249

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

OS/9, FLEX, COCO, UNIFLEX SOFTWARE

SUPER SLEUTH DISASSEMBLER   EACH $99-FLEX, $101-OS/9, $100-UNIFLEX
interactively generates source on disk with labels, includes xref, label definition, binary file editing, etc.
specify 6800,1,2,3,5,8,9/6502 version or Z-80/8080/85 version

(OBJECT ONLY)   EACH $50-FLEX & OS/9, $49-COCO DOS
COCO DOS available in 6800,1,2,3,5,8,9/6502 version only

CROSS-ASSEMBLERS   EACH $50-FLEX, $55-OS/9, $60-UNIFLEX, ALL $100
specify for 6800/1, 6502, 6805, Z-80, or 8080/48/85
OS/9 version requires Microware RMA or FHL OSM macro assembler
FLEX version requires TSC ASMB or FHL ASM or OSM macro assembler

DEBUGGING SIMULATORS   EACH $75-FLEX, $100-OS/9, $80-UNIFLEX
specify 6800/1, 6805/146805, 6502, or 16809 OS/9 only)

OS/9 AND UNIFLEX SIMULATORS   EACH $100-FLEX
debug OS/9 and UNIFLEX application programs under FLEX using TSC DEBUG facility

6502 TO 6809 ASSEMBLER TRANSLATOR $75-FLEX, $85-OS/9, $80-UNIFLEX
translates 6502 programs to 6809, noting inexact conversions

6800 TO 6809 & 6809 PIC TRANSLATORS $50-FLEX, $75-OS/9, $60-UNIFLEX
translates 6800 programs to 6809, 6809 programs to PIC

FULL-SCREEN FLEX AND UNIFLEX TSC XBASIC PROGRAMS FOR 6809
(with complete cursor control)

DISPLAY GENERATOR/DOCUMENTOR	$50-FLEX, $75-UNIFLEX
MAILING LIST SYSTEM	$100-FLEX, $110-UNIFLEX
INVENTORY WITH MRP	$100-FLEX, $110-UNIFLEX
TABULA RASA SPREADSHEET	$100-FLEX, $120-UNIFLEX

DISK UTILITY PROGRAM LIBRARY $50-FLEX
edit disk sectors, sort directory, maintain master catalog, etc.

CMODEM TELECOMMUNICATIONS PROGRAM $50-FLEX & OS/9 & UNIFLEX
provides menu-driven telecommunications facilities, with terminal mode, up/down load, MODEM7 protocol, etc.

5.25" SOFT-SECTORED DISKETTES   EACH SET OF 50 $75-SSDD, $85-DSDD
with Tyvek jackets, hub rings, labels

Specify operating system, computer make and type, terminal type
Programs provided in source form on diskette, specify size and density
Contact CSC for full catalog and dealer info, printed manuals provided with products.
For VISA and MASTER CARD, give account, exp. date, phone. US funds only.
Add 5% for shipping software, but not for diskettes.
(UNI)FLEX trademark Technical Systems Consultants   OS/9 trademark Microware

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-1717/4570

SOFTWARE FOR THE HARDCORE

.. FORTH PROGRAMMING TOOLS from the 68XX&X ..
.. FORTH specialists — get the est!! ..

NOW AVAILABLE — A variety of rom and disk FORTH systems to run on and/or do TARGET COMPILATION for

6800, 6301/6801, 6809, 68000, 8080, Z80

Write or call for information on a special system to fit your requirement.

Standard systems available for these hardware—

EPSON HX-20 rom system and target compiler
6809 rom systems for SS-50, EXORCISER, STD, ETC.
COLOR COMPUTER
6800/6809 FLEX or EXORCISER disk systems.
68000 rom based systems
68000 CP/M-68K disk systems, MODEL II/12/16

tFORTH is a refined version of FORTH Interest Group standard FORTH, faster than FIG-FORTH. FORTH is both a compiler and an interpreter. It executes orders of magnitudes faster than interpretive BASIC. MORE IMPORTANT, CODE DEVELOPMENT AND TESTING is much, much faster than compiled languages such as PASCAL and C. If Software DEVELOPMENT COSTS are an important concern for you, you need FORTH!

firmFORTH™ is for the programmer who needs to squeeze the most into roms. It is a professional programmer's tool for compact rommable code for controller applications.

™ tFORTH and firmFORTH are trademarks of Talbot Microsystems
™ FLEX is a trademark of Technical Systems Consultants, Inc
™ CP/M-68K is trademark of Digital Research, Inc

tFORTH™
from TALBOT MICROSYSTEMS
NEW SYSTEMS FOR
6301/6801, 6809, and 68000

---> tFORTH SYSTEMS <---

For all FLEX systems: GIMIX, SWTP, SSB, or EXORcisor Specify 5 or 8 inch diskette, hardware type, and 6800 or 6809.

.. tFORTH — extended fig FORTH (1 disk) $100 ($15)
with fig line editor.
.. tFORTH+ — more! (3 5" or 2 8" disks) $250 ($25)
adds screen editor, assembler, extended data types, utilities, games, and debugging aids.
.. TRS-80 COLORFORTH — available from The Micro Works
.. firm FORTH — 6809 only. $350 ($10)
For target compilations to rommable code.
Automatically deletes unused code. Includes HOST system source and target nucleus source. No royalty on targets. Requires but does not include tFORTH+.
.. FORTH PROGRAMMING AIDS — elaborate decompiler $150

.. tFORTH for HX-20, in 16K roms for expansion unit or replace BASIC $170

.. tFORTH/68K for CP/M-68K 8" disk system $290
Makes Model 16 a super software development system.

.. Nautilus Systems Cross Compiler
— Requires: tFORTH + HOST + at least one TARGET:
— HOST system code (6809 or 68000) $200
— TARGET source code: 6800-$200, 6301/6801—$200
same plus HX-20 extensions— $300
6809—$300, 8080/Z80—$200, 68000—$350

Manuals available separately — price in ( ).
Add $6 system for shipping, $15 for foreign air.

TALBOT MICROSYSTEMS   1927 Curtis Ave., Redondo Beach, CA 90278 (213) 376 9941

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine $1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 Package, besides the price, is the DOCUMENTATION. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products wae the restrictions placed on releasing that information by Microsoft: I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is now available for the "little old Color Computer": OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the 68000, also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9; the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "Official Radio Shack" version for Review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3030), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

FREE SAMPLE ISSUE

1-800-338 6800

MON.-FRI. 9-5 E.S.T.

USA-$12.50 per year. Canada& Mexico-$19.50 per year

Surface Foreign-$24.50 per year. Airmail Foreign-$48.50 per year

Color Micro Journal™

TM Color Micro Journal is a trademark of Computer Publishing Inc.

5900 Cassandra Smith Rd.
Hixson,TN. 37343

NEW OS-9® SOFTWARE

HELP

User-expandable generic help facility

- Includes data for online help with OS-9 utilities
- Fast, efficient disk storage
- Three levels of nesting
- Wild Card searching
- Automatic display of available help
- Steps the user until he finds the answer

DISK BACKUP

Controlled hard disk-to-floppy backup with restore capability

- Handles files larger than output media
- Single file, Wild Card search, current directory only, current-and-all-subdirectories
- Date and time for incremental backup
- Operator-friendly, handles error conditions smoothly
- Use to create optimized disks

TERMINAL

Communications program for OS-9

- Use your micro as an intelligent terminal
- Go online over phones or connect directly
- Transfer data in both directions
- Menu-driven
- XON/XOFF support required — BASIC09/RUNB required

DO WE HAVE YOUR NAME & ADDRESS
For new products news & announcements?

see us at PRINCETON

JBM'S MIDWARE

VISA

YES! I want OS-9 SOLUTIONS!

Please send me:

HELP ($69.00)	________
DISK BACKUP ($99.00)	________
TERMINAL ($95.00)	________
PA Sales Tax 6%	________
Postage/Handling $5	________
TOTAL	________

□ I'm interested, need more information

SHIP TO: ________________

____

____

Make Check/Money Order payable to: The JBM Group, Inc.
Or by VISA/MASTERCARD
Acct. # ________________
Exp. date: ________________
Signature: ________________

Required Distribution: CoCo (Y/N): ______

5¼ 35 track	________
5¼ 40 track	________
5¼ 80 track	________
8" 77 track	________

ORDER FROM

Dept. 6812

the JBM group

The JBM Group, Inc.

Continental Business Center
Front & Ford Streets
Bridgeport, PA 19405

Tel: 215-337-3138/TWX: 510-660-3999

* OS9 is a registered trademark of Microware Corp.

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display procesor. High resolution 256 x 192 pixel display with 15 colors. 16K Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation. External Video input allows subtitling. NTSC composite video output.

SOUND EFFECTS AND MUSIC
- Three AY3-8910 Programmable Sound Generators
- Nine simultaneous voices
- Three independent noise sources
- Onboard stereo amplifier drives two 8 ohm speakers

ADDITIONAL I/O CAPABILITIES
- Eight analog inputs with 8 bit resolution
- Suppo s four joysticks with ushbut ion switches
- Eighi bit parallel I/O t
- Entire unil maps into 256 bytes of memory

FBASIC

TERMINUS DESIGN INC, in conjunction with Microware Systems Corporation, is proud to announce FBASIC an enhancement of Microware's 6800/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions. Disk I/O and fast integer arithmetic with multiple-precision capability Graphics verbs and functions fully support the Arcade 50.

A CADE 50 assembled and t sled	$325 00
Video and Audio conn ctor set	15 00
4 Joystick connector set	15 00
2 Radio Shack Joysticks	24 00
Gold Molex connectors	12 00
A/BASIC for 6800	110 00
FBASIC for 6 09	110 00
FBASIC (with ARCADE 50)	75 00
ARCADE 50 RGB	375 00
LABVIDEO (Motorola EXORbus)	375 00
NEW M 09 8809 Processor Board	225 00
256K Dynamic Memory Board	795 00
256K Dynam c Memory Board (w/64K)	395 00
64K Dynamic Memory Bo rd	295 00

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4866

TERMS: CASH, VISA, MC, COD

OS9
APPLICATION SOFTWARE

ACCOUNTS PAYABLE $349	GENERAL LEDGER with CASH JOURNAL $449	PAYROLL $549
ACCOUNTS RECEIVABLE $349		SMALL BUSINESS INVENTORY $349

COMPLETE DOCUMENTATION $19.95

OS9 & BASIC 09 ARE TRADEMARK OF
MICROWARE, INC. & MOTOROLA CORP.

SPECIALTY
ELECTRONICS

(405) 233-5564
2110 W. WILLOW — ENID, OK 73701

Computer Servo Controlled Robot Arm

Call or Write for Free Catalog

Robot-1

Keyboard or Joystick Control

Remembers Everything It Did & does it again

Typical System Includes:
- Robot-1 & Cables
- 6 Channel Servo Controller
- Power Supply
- All Software with source code

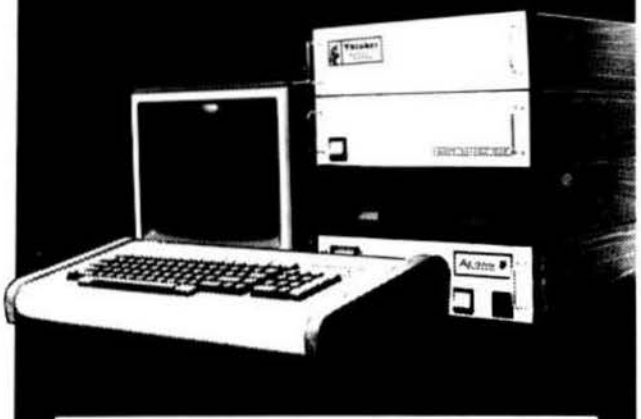
Modular Robotic Accessories:
- Mobile Cart for Traveling Robot
- Radio Links between all Functions
- Robot-mounted MicronEye
- Ultrasonic Range Finder

Robot-1 Series
starting at $289.00
for the Color Computer
and 6809 SS50 Computers

Additional Systems Available
Robot-1R for Radio Control Systems

Analog Micro Systems

5660 Valmont Road . Boulder, Colorado 80301 . Tel: (303) 444-6809



68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

6809 COMPLETE SYSTEM 6809

ADVANCED TECHNOLOGY ECONOMICAL

The PT69 COMPLETE SYSTEM features the proven PT69 single board computer. Powerful performance + reliability - an unbeatable combination! Complete Systems feature:

- Double Sided/Double Density Drives (40 or 80 track) Cabinet, Power Supply
- Optional CRT and Printer
- 1MHz 6809E Processor
- 2 RS232 Serial Ports
- 2 8-Bit Parallel Ports
- 56K RAM: 4K EPROM: 4K I/O

* COMPLETE SYSTEM with PT69 Board, 2 DS/DD 5¼" 40 Track Drives, Cabinet, and Power Supply	$999.95
* PT69, Assembled, Tested, with Power Supply and Cabinet.	$399.95
* PT69, Assembled and Tested	$299.95

— Kits are available —

— Write for information on CRT and Printer —

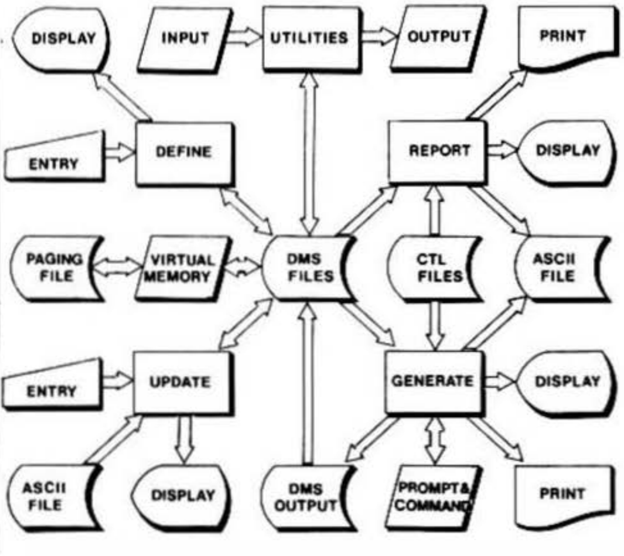
OEM Inquiries
Welcome

PERIPHERAL TECHNOLOGY
"Supplying Your Computer Needs Since 1978"
3760 Lower Roswell Road
Marietta, Georgia 30067

VISA/MASTERCARD/CHECK/COD 404/973-0042

XDMS

Data Management System



System Architecture

WESTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VMOEX utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system, and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I $129.95

XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II $199.95

XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file format, graphic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS Level III $269.95
XDMS System Documentation only ($10. credit toward purchase). . .$ 24.95

XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus the general ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products (or services), transactions, invoicing, etc. Easily configured to most environments.
XACC General Accounting System (Requires XDMS, pref. Lv. III). . $299.95
XACC System Documentation only ($10. credit toward purchase). . .$ 24.95

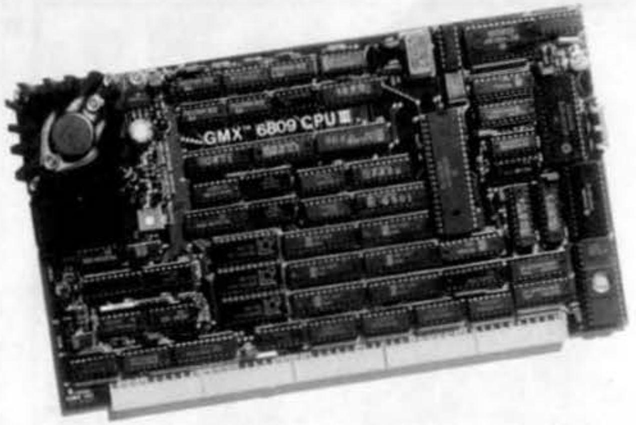
WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S.
Terms: Check, Money Order, Visa or Mastercharge. Shipment first class.
Add P&H $2.50 ($7.50 Foreign). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. MEDIA, 1-800-338-6800, Consultation: 914-941-3552 (evens).

FLEX is a trademark of Technical Systems Consultants, Inc

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

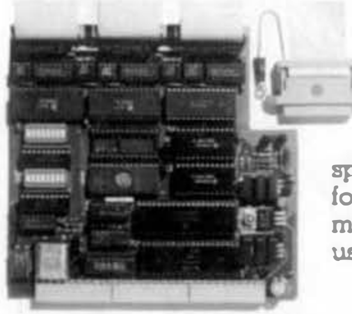
GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.

The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business: whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

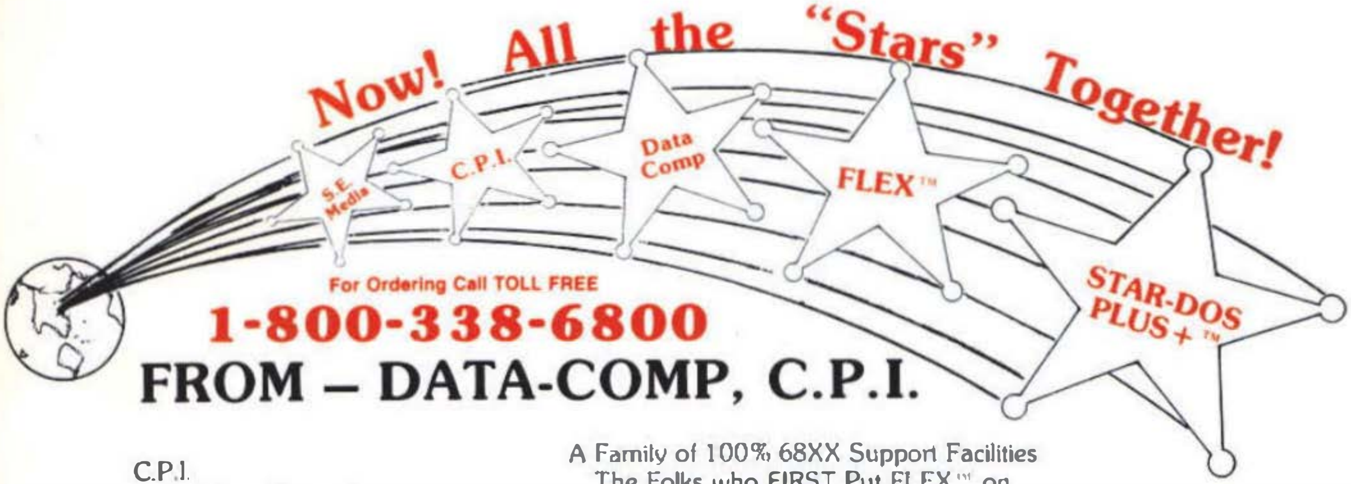
Please phone or write if you need further information.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX inc.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Now! All the "Stars" Together!



For Ordering Call TOLL FREE

1-800-338-6800

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.⁰⁰ **Only $79.⁹⁵**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.⁹⁵

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.⁹⁵

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

TSC Editor
Reg $50.00
NOW $35.00

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ Super 800 Support
+ Free Color Micro Journal 1 yr. sub.
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Assembler
Reg $50.00
NOW $35.00

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER,
POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT **J&M, OR RADIO SHACK.**

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$659.95
COLOR COMPUTER II 64K W/EXT. BASIC	$189.95

CONTROLLERS

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC. SPECIFY WHAT DISK BASIC.	$139.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

DISK DRIVE CABLES

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

MISC

64K UPGRADE W/MOD. INSTRUCTIONS, C,D,E,F, AND COCO 2	$ 49.95
HJL KEYBOARDS	$ 69.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 29.95
RADIO SHACK DISK BASIC 1.1	$ 29.95
RADIO SHACK EXT. BASIC	$ 39.95
SCREEN CLEAN CLEARS UP VIDEO DISTORTION	$ 39.95
MEMOREX DISKS 5" SS,DD	$ 24.00
SHIPPING INCLUDED ON DISK PRICES	
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$650.00
EPSON FX-100	$799.00
EPSON FX-80	$549.00
EPSON MX-70	$200.00

SERIAL BOARDS FOR PRINTERS

MX-SERIES	$119.95
FX-SERIES	$ 99.95

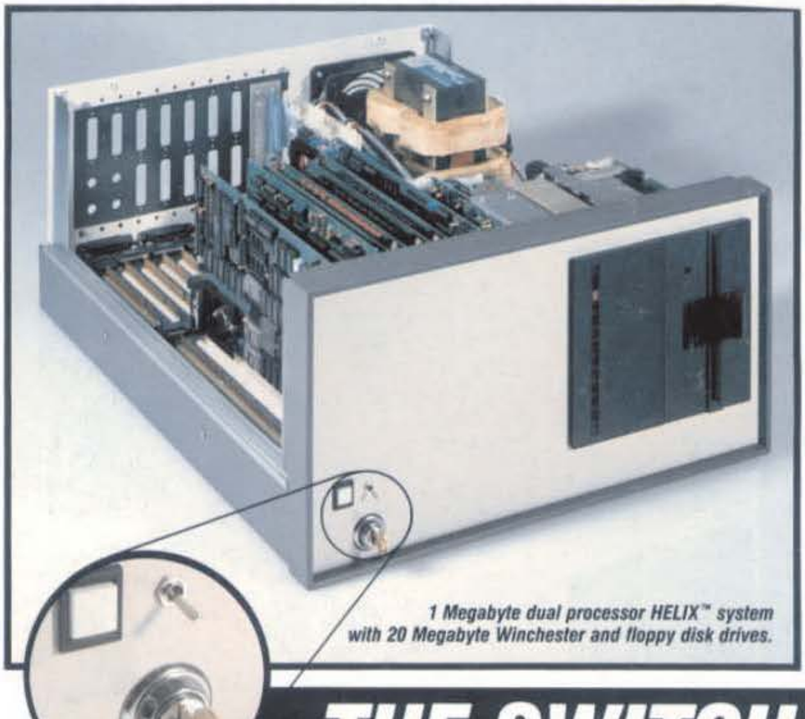
USA ADD 2% SHIPPING
FOREIGN ADD 5% SHIPPING

5900 Cassandra Smith Rd. Hixson, TN 37343

*FLEX is a Trademark of Technical System Consultants
*STAR-DOS+ is a Trademark of STAR-Kits & Data-Comp

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025



1 Megabyte dual processor HELIX™ system with 20 Megabyte Winchester and floppy disk drives.

ONCE AGAIN HAZELWOOD COMPUTER SYSTEMS demonstrates its leadership in computer technology by delivering the only computer system capable of switching between either the 6809 or the 68000 processor. Switching is easily accomplished by a simple front panel toggle switch. The reason we can offer this exclusive feature now, is that when our proven 6809 processor board was designed several years ago, we had the foresight to include the bus controls that allow processor switching.

Hazelwood Computer Systems is also proud to be the first S-50/S-64 bus manufacturer to license and deliver the OS9/68K Operating System from Microware Systems Corporation. OS9/68K is the 68000 version of the popular and powerful OS9 Operating System. Utilizing our proven MC-20 disk controller, OS9/68K can conveniently share a Winchester disk with OS9. Changing from 6809 to 68000 operation is as simple as switching processors and booting the new system from the Winchester disk.

The ease of switching processors and operating systems makes a HELIX™ dual processor system the natural choice for software development. In addition, the advanced design of HELIX™ equipment, emphasizing performance and reliability, makes HELIX™ boards and systems the best value in computing offered anywhere.

System prices vary with configuration. Call for exact pricing.

THE SWITCH IS ON...

The CP-08 processor board utilizes a 68008 processor running at 10 Mhz clock rate. Using proprietary bus synchronization circuitry and single cycle DMA, the CP-08 achieves a marked performance increase over a 2 MHz 6809. Offering absolute compatibility with the 68000 instruction set, the 68008 addresses up to 1 Megabyte of memory. Also included on the CP-08 are up to 4K of ROM, an interrupt timer, and, with battery backup operation, a clock/calendar and 2K RAM. Implemented as a standard S-50 board, the CP-08 brings 68000 operation to S-50 bus computers.

ORDER: CP-08 **PRICE: $595**

The MC-20 Mass Storage Controller board interfaces up to 4 floppy and 8 Winchester disk drives to the S-50/S-64 bus. The MC-20 is an intelligent controller with its own 2 Mhz 6809 processor and 56K RAM. It provides DMA data transfers to a full 24 bit address. All disk operation requests are by logical block number, with the controller performing the necessary track/sector address calculations. Any combination of 5¼ or 8 inch floppy drives can be accommodated with all drive parameters, such as write precompensation, software controlled for each individual drive. Winchester drives are connected via a SASI bus interface. Block address mapping is provided which allows a single drive to be segmented into several logical units. The MC-20 is the controller of the MS-20 Mass Storage Subsystem which includes a 20 Megabyte Winchester drive.

ORDER: MC-20 **PRICE: $695**

OS9/68K

ORDER: OS9/68K **PRICE: $250**

OS9/68K offers increased performance and larger user memory space while retaining all of the features of OS9. Disk file compatibility and operational similarity assures that present OS9 users can easily transfer their operations to the 68000. Included are an editor, assembler, linker, and debugger. A C compiler is available now. BASIC09 and other languages will be avilable soon.

All items available stock to 30 days.
Prices subject to change without notice.

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™

OS9 and OS9/68K are registered trademarks of microware Systems Corp. HELIX is a trademark of Hazelwood Computer Systems.